AGH University of Krakow

FIELD OF SCIENCE ENGINEERING AND TECHNOLOGY

SCIENTIFIC DISCIPLINE AUTOMATION, ELECTRONICS, ELECTRICAL ENGINEERING AND SPACE TECHNOLOGIES

DOCTORAL DISSERTATION

Decision-Making and Risk-Aware Navigation in Mobile Robotics: From Path Planning to Emergency Response

Author: Mehmet KARA

Supervisor: prof. dr hab. Andrzej Maciej Skulimowski

Completed at: AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering

Kraków, 2025

I would like to express my deepest gratitude to my supervisor, whose guidance, patience, and constructive criticism have shaped not only this dissertation but also my way of thinking as a researcher. I am also grateful to the members of the faculty and colleagues at AGH, whose insights and discussions enriched my academic journey. Special thanks go to my family and my beloved spouse, who stood by me with endless support, encouragement, and understanding throughout this demanding process. Their trust gave me strength in moments of doubt and reminded me that perseverance is always worthwhile. Without their love, this work would never have reached its completion. Although my dissertation is not directly about climate change or sustainability, I cannot separate my scientific identity from my personal convictions. As someone who dreams of a greener world and feels a responsibility to act against global warming, I believe that every piece of research; whether in robotics, artificial intelligence, or engineering; can ultimately serve humanity's effort to create a more sustainable future. It is my hope that the knowledge and methods explored here will one day find applications that contribute, even indirectly, to the preservation of our planet. To all who have supported me, challenged me, and inspired me on this path,
thank you . . .

Abstract

In modern robotic navigation, efficiency is no longer defined solely by the optimality of the path but by the robot's ability to make context-based decisions. This dissertation investigates the trade-offs between avoiding obstacles and tolerating controlled risks to improve operational performance in autonomous mobile robots. Although conventional systems treat all obstacles as hazards to be avoided, this work introduces a learning-based framework that enables robots to estimate potential collision damage and determine whether avoidance is truly necessary.

The methodology integrates supervised learning models trained on simulated collision data and couples them with classical path planning algorithms. The result is a system capable of adapting navigation behavior based on obstacle type, potential risk, and mission context. This hybrid approach not only minimizes unnecessary detours but also conserves energy and time, resources that are critical in complex environments.

A key contribution of this thesis is the explicit integration of damage-aware learning into navigation, filling a gap in the literature where obstacle handling has traditionally been risk-blind. Furthermore, the thesis proposes a flexible emergency planning framework designed to be adapted across various robotic domains. The framework emphasizes preparedness and operational continuity in unpredictable situations. Experimental validation, including multiple scenario simulations and statistical analysis, confirms the effectiveness of the proposed methods in improving both safety and efficiency.

This study enhances robotic autonomy in unpredictable and changing conditions by integrating conventional algorithms with data-centric risk assessments, providing effective tools for practical applications in fields such as industrial logistics, agriculture, and disaster response.

Streszczenie

W współczesnej nawigacji robotów mobilnych efektywność nie jest już definiowana wyłącznie przez optymalność trasy, lecz przez zdolność robota do podejmowania decyzji w oparciu o kontekst. Niniejsza rozprawa analizuje kompromisy między omijaniem przeszkód a akceptacją kontrolowanego ryzyka w celu poprawy wydajności operacyjnej autonomicznych robotów mobilnych. Podczas gdy konwencjonalne systemy robotyki mobilnej traktują wszystkie przeszkody jako zagrożenia, które należy omijać, w niniejszej pracy zaproponowano nowe podejście oparte na uczeniu maszynowym, umożliwiające robotom oszacowanie potencjalnych szkód wynikających z kolizji i określenie na tej podstawie, czy unikanie przeszkód jest rzeczywiście konieczne.

Zaproponowana metodologia łączy modele uczenia nadzorowanego, wytrenowane na podstawie symulowanych danych kolizyjnych, z klasycznymi algorytmami planowania ścieżek. Wynikiem jest system zdolny do adaptacji zachowania nawigacyjnego w zależności od rodzaju przeszkody, potencjalnego ryzyka i kontekstu misji. Hybrydowe podejście nie tylko minimalizuje niepotrzebne objazdy, ale również oszczędza energię i czas, które są kluczowe w złożonych zadaniach.

Istotnym wkładem tej rozprawy jest bezpośrednia integracja uczenia opartego na oszacowaniu potencjalnych uszkodzeń w procesie nawigacji, co wypełnia lukę w literaturze, w której, z wyjątkiem nielicznych wyników, pokonywanie przeszkód było pozbawione oceny ryzyka. Ponadto zaproponowano elastyczne ramy planowania awaryjnego, możliwe do adaptacji w różnych dziedzinach robotyki. Ramy te podkreślają znaczenie gotowości i ciągłości operacyjnej w nieprzewidywalnych sytuacjach. Walidacja eksperymentalna, obejmująca wiele scenariuszy symulacyjnych oraz analizy statystycznej, potwierdza skuteczność proponowanych metod w zakresie poprawy zarówno bezpieczeństwa, jak i efektywności.

Niniejsza praca stanowi również wkład do teorii autonomii robotów w nieprzewidywalnych i dynamicznie zmieniających się warunkach poprzez integrację klasycznych algorytmów planowania ścieżek z oceną ryzyka opartą na danych obserwacyjnych, dostarczając praktycznych narzędzi dla zastosowań w logistyce przemysłowej, rolnictwie i reagowaniu kryzysowym.

Contents

List of Figures

List of Symbols

Symbol	Description
$\mathcal{O}$	Set of obstacles in the environment
o_i	i-th obstacle in $\mathcal{O}$
$\mathcal{P}$	Continuous or discrete path in the environment
p_j	j-th point in a discrete path $\mathcal{P}$
$D(o_i)$	Damage cost function for obstacle o_i
$\mathcal{E}$	Set of emergency scenarios
e_i	i-th emergency scenario
s_i	Severity level of e_i
r_i	Required response time for e_i
c_i	Affected component(s) in e_i
f	Decision function mapping obstacles to actions
$T(f_j, e_i)$	Time required to execute response f_j in e_i
$R(f_j, e_i)$	Residual risk after executing f_j in e_i
$C(f_j, e_i)$	Resource/energy cost of executing f_j
α, β, γ	Weights in multi-objective optimization

Chapter 1

Introduction

Autonomous mobile robots have found applications in logistics, inspection, disaster response operations, and many other fields. The critical component for these tasks is path planning [153]. Traditional approaches to path planning have focused on geometric feasibility and efficiency. However, real-world applications highlight the importance of additional objectives, such as time, energy, and safety risks, which are often addressed implicitly or only after the fact [173]. Most studies model obstacles as hard constraints, which can increase the detour length. In contrast, another line of work investigates traversability and risk-aware motion involving contact or soft interactions [94, 171]. Moreover, standardized frameworks for emergency handling in robotics remain limited; existing approaches are often tailored to specific domains.

Many comparative studies emphasize shortest paths and collision-free motion, while time and energy aspects are often simplified or decoupled. However, as environments become more dynamic, reaching a goal is insufficient. Planners must reason over multiple objectives, such as time, energy, and safety, rather than assuming equal severity for all obstacles [178].

A second gap concerns emergency handling in robotic environments. Given the diversity of platforms and missions, a single prescriptive protocol is rarely effective [171, 181]. We propose and evaluate a configurable emergency-response framework designed to adapt to application-specific constraints.

This dissertation studies *end-to-end planar* path planning for *unmanned ground vehicles (UGV)* and discusses how navigation quality improves when deviation costs (*time/energy*) and potential damage are *explicitly* traded off within a formal decision-making optimization (MO) framework. Concretely, we compare the extra cost of avoiding perceived obstacles with a learned estimate of damage in the event of a collision and select actions accordingly. The proposed method models risk as a quantitative parameter in the optimization process, allowing its impact to be systematically evaluated. This shift facilitates making informed decisions that more closely align with operational goals [175, 177].

In summary, this work focuses on two underrepresented gaps in mobile robotics:

- (i) Extra time/energy trade-offs when minor obstacle contact may be acceptable, using a learning-based damage estimator;
- (ii) An adaptable emergency-response framework that can be instantiated for distinct UGV applications.

To address these gaps, the following outlines the systematic approach of our work:

1. **Introduction** (*Scope of the Thesis*): This section introduces the current landscape of autonomous mobile robotics and outlines the main research objectives. This also explains why the topics are important in light of existing studies.

2. **Literature Review** (*Previous Work*): We reviewed past studies that focused on path planning, decision-making, learning, damage estimation, and obstacle avoidance. We then discussed how those aspects could be combined more effectively.

3. **Machine Learning Advances** (*Second Part of the Literature*): This part focuses on how machine learning has been used in mobile robotics. We explored recent developments and selected the techniques that are most relevant to this work. Owing to the volume of work, we present machine-learning advances in a dedicated chapter.

4. **Evaluation of Algorithms** (*First Part of the Methodology*): To find the shortest path, we examined and compared popular algorithms using simulation tools. This comparison constitutes the first part of our methodology.

5. **Methodology** (*What We Did*): We describe how we combined path planning with damage estimation. We trained a model using the data we created, allowing the robot to predict potential damage and adjust its path accordingly.

6. **Emergency Framework** (*Second Contribution*): In addition to the main methodology, we propose a flexible emergency response framework for robotic environments. This framework is designed to adapt to different conditions and needs.

7. **Experiments** (*Testing and Results*): In this section, we test the proposed approach and evaluate how it performs in various scenarios, including emergencies. We concluded the thesis with a summary of the findings.

8. **Summary** (Conclusion, Dissemination, and Ethics): This is the summary of the dissertation, and the overall thesis is concluded. Future work is mentioned, along with ethical considerations and dissemination through related publications.

To ensure that each phase was thorough, we published an article corresponding to each major part of the study. Fig. 1.1 summarizes the overall structure of the thesis and illustrates how the chapters are related.

Thesis formulation. We formulate robot path planning as an explicit trade-off between deviation cost (time/energy) and the expected damage from obstacle contact. Instead of the common avoid-all-obstacles paradigm, the planner decides whether to avoid or traverse an obstacle by minimizing a risk-aware objective that includes both efficiency and damage. This broadens the applicability of path planning methods in cluttered, real-world environments.

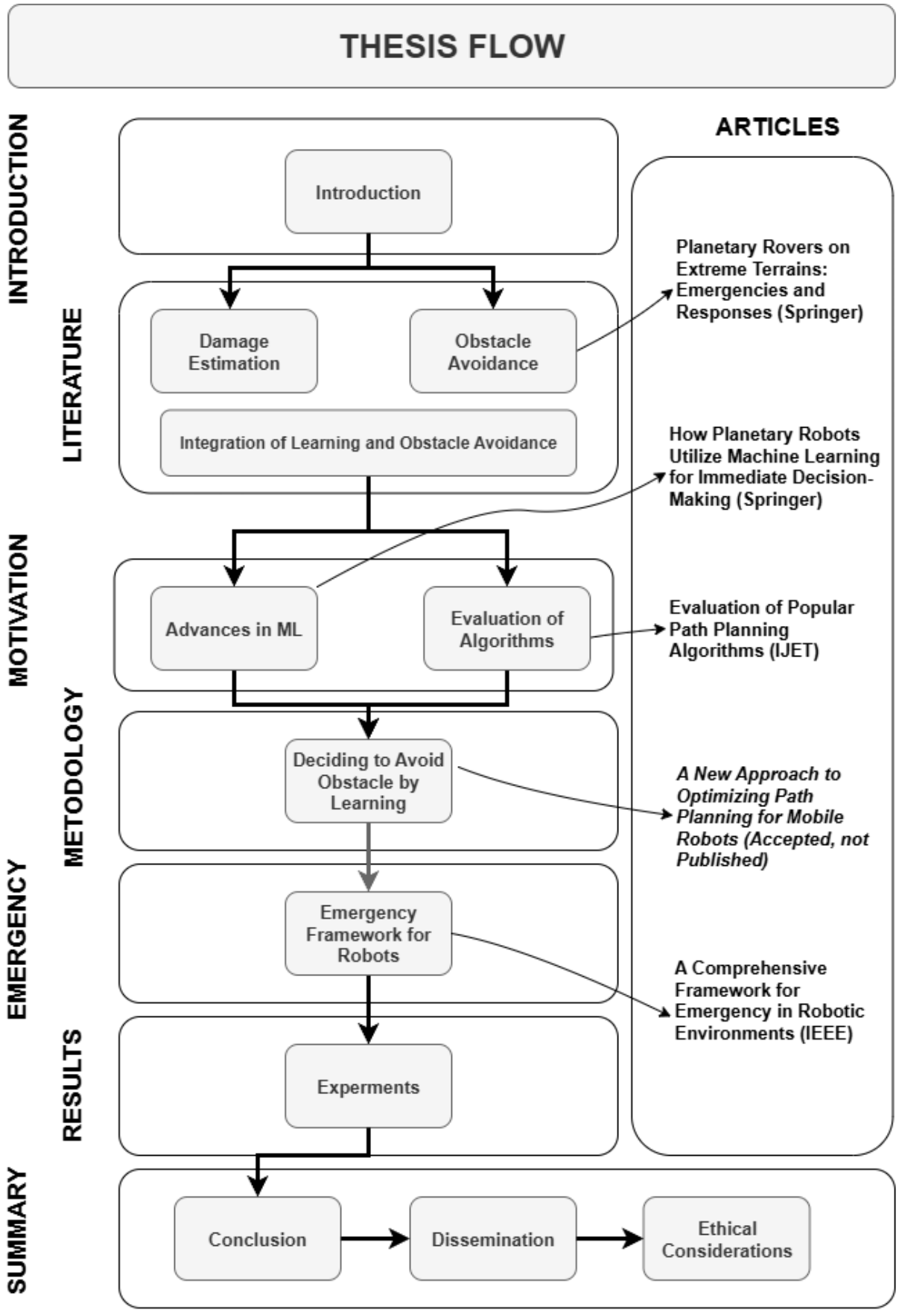


Figure 1.1: Thesis flow diagram.

Thesis

This dissertation investigates *end-to-end path planning on the plane for UGV* under multiple, sometimes conflicting objectives. The central idea is that navigation quality improves when robots explicitly consider not only the geometric feasibility of a path but also the trade-offs between additional travel costs and the potential damage from interacting with obstacles.
The contribution of this work lies in integrating three key components into a single framework:

1. learning models to estimate obstacle-specific damage levels,
2. decision functions that weigh path efficiency against predicted risks,
3. a coordination layer enabling multi-robot systems to respond effectively to emergencies.

The thesis posits that shifting from the traditional *avoid-all-obstacles* paradigm toward a risk-aware approach can enhance both efficiency and resilience. This hypothesis is examined through simulation studies that evaluate when and how such trade-offs yield measurable benefits for autonomous ground robots.

The Introduction is organized into four main parts. First, we present the **background** of mobile robots and the importance of obstacle avoidance. Then, we define the **problem** that this research seeks to address. The third part outlines the **objectives** of the study. Finally, we explain the general **significance** of the proposed approach.

1.1 Background

Autonomous Mobile Robots: Autonomous mobile robots integrate advanced perception, localization, and control systems to achieve independent operation without human intervention [176]. These systems rely on real-time sensor fusion, autonomous decision-making algorithms, and environment-aware navigation. Equipped with sensors and processors, these robots can perceive their environment and navigate within complex settings [230]. They have found applications in various industries, including manufacturing, logistics, healthcare, and agriculture. That is, the usage of robots is increasing steadily as a result of the evolving demands of human life [218].

These systems use artificial intelligence, sensor technologies, and mobility to perform tasks in structured and unstructured environments. From factories to homes and outdoor locations, they can adapt to different surroundings. Sensors such as lidar, cameras, and ultrasonic devices help collect real-time data, which is used to guide actions and respond to changes [49].

One of their defining features is the ability to autonomously navigate unknown environments. Techniques such as simultaneous localization and mapping (SLAM) allow robots to build maps of their surroundings while simultaneously determining their own position within them [243]. SLAM provides the state estimate and environment representation, whereas *PP* consumes this information to compute feasible trajectories. Representative examples of SLAM applications can be found in warehouses and dynamic outdoor settings [245].

As mentioned above, the range of applications for ground-based mobile robots continues to expand. In industry, warehouse operations are supported by UGV that transport goods, track inventory, and improve order fulfillment. In agriculture, wheeled and tracked UGV are increasingly used for precision farming tasks such as crop monitoring and targeted spraying [184]. In healthcare, mobile service robots support logistics inside hospitals, while in search-and-rescue scenarios, ground robots operate in cluttered and partially collapsed environments [195]. With technology becoming more affordable, UGV are expected to be increasingly integrated into operations, improving productivity and safety across sectors [166].

Traditional Approaches to Path Planning: Traditional path planning techniques have formed the backbone of mobile robotics for several decades. *Dubins curves* remain a standard tool for modeling curvature-constrained trajectories of nonholonomic vehicles [133]. Graph-based approaches, such as *Voronoi diagrams* and visibility graphs, provide geometric frameworks that enable efficient shortest-path computation in continuous environments [147]. More recently, sampling-based methods, including the *Probabilistic Roadmap (PRM)* and *Rapidly-Exploring Random Trees (RRT)*, have extended their applicability to high-dimensional configuration spaces by trading exact completeness for probabilistic guaranties and tractable runtime [153].

In this dissertation, these traditional algorithms are treated as *baselines*. They reliably generate feasible paths under basic constraints, but they do not capture the explicit trade-off between 'the time and energy cost of avoiding' and 'potential collision damage'. *Our contribution extends these foundations with a risk-aware optimization layer and a real-time decision mechanism that incorporates risk assessment into the planning process.*

Beyond these standard form formulations, two variants are particularly relevant for situating the scope of this work:

- *Coverage path planning (CPP)* generates trajectories that allow a robot to systematically traverse an environment and cover its entire area. Applications include autonomous vacuum cleaning, agricultural field monitoring by UGV, and structured inspection tasks [217].
- *Multi-target path planning (MTPP)* extends the single-source-to-goal paradigm by requiring sequential visits to multiple targets. Solutions often rely on adaptations of the traveling salesman problem or vehicle routing heuristics [54].

Positioning this dissertation against such established categories clarifies its novelty: whereas classical approaches focus on feasibility and efficiency, our emphasis lies in risk-aware decision making. To ensure coherence with the targeted platform, all obstacle models and evaluations are explicitly framed with respect to UGV geometry, dynamics, and operational context.

For example, a low static barrier may be critical for wheeled UGV but negligible for legged platforms, while soft ground conditions present challenges unique to UGV locomotion [94]. By anchoring the analysis to UGV characteristics, the evaluation criteria remain consistent, and the proposed contributions can be rigorously assessed.

Obstacle Avoidance: One of the main challenges facing mobile robots is dealing with obstacles along their path. These obstacles may be static or dynamic and can interfere with the robot's ability to complete its

tasks [222]. Navigating around them effectively is essential for maintaining both safety and efficiency. With the growing deployment of mobile robots in dynamic environments such as warehouses and urban settings, robust obstacle avoidance strategies have become critical for safe operations [154].

Obstacle avoidance is critical to ensuring that robots can operate safely in unpredictable settings. By detecting and responding to obstacles, robots can avoid collisions, minimize damage, and continue operating without interruption. This is especially vital in areas such as autonomous vehicles, industrial robotics, and service robots in crowded environments. These tasks rely on accurate sensors such as lidar, cameras, and ultrasonic systems, along with algorithms that interpret the data and generate movement instructions [247, 114].

In more complex environments, in some contexts, robots must go beyond simple avoidance and consider traversing or negotiating obstacles [183]. This is especially true in outdoor or unstructured areas, or when dealing with objects of irregular shape or size indoors. In such cases, robots must assess the obstacle, choose a path, and perform precise movements to avoid or traverse it. In addition, special equipment, such as articulated arms or manipulators, may be needed to clear a path or interact with the obstacle [185].

The ability to avoid and overcome obstacles not only improves safety but also expands the environments in which robots can work effectively [29]. Whether navigating tight spaces in a warehouse, rough ground in rescue missions, or traffic in urban settings, robots with advanced obstacle handling skills can perform more reliably. As robotics continues to evolve, improvements in sensing and decision-making will make these capabilities even more robust, allowing robots to operate with greater independence in complex situations [122, 245].

Learning-Based Damage Estimation: A cornerstone of the proposed risk-aware framework is the ability to quantitatively predict the potential damage resulting from a collision with an obstacle. This is achieved through a machine learning model trained on obstacle features such as size, material, and mobility. In this work, we employ a *Multiple Linear Regression (MLR)* model to estimate the continuous damage score. The relationship between the obstacle features (predictors) and the damage (target variable) is formulated as shown in Eq. 5.11 in Chapter 5. This model allows the robot to make informed, real-time decisions by comparing the predicted cost of traversal against the cost of avoidance, thereby operationalizing the core trade-off at the heart of this thesis.

Emergency for Multi-Robot Systems: Operating multi-robot systems under emergencies introduces complex challenges for safety and operational continuity [108, 179]. While these systems enhance productivity and capabilities, their integration into dynamic real-world environments also amplifies potential risks. Unexpected failures, environmental hazards, or internal malfunctions can disrupt coordinated missions, leading to cascading failures that threaten both the robots and their operational objectives [175].

Given the substantial investment embodied in robotic platforms and the paramount importance of human safety, the development of robust emergency strategies is essential [237]. However, the variability in robot design, mission context, and environmental conditions renders rigid, standardized emergency protocols largely ineffective [108]. This is particularly true in multi-robot operations, where emergency manage-

ment is not merely about individual unit safety but also involves maintaining team-level coordination and fulfilling collective mission goals despite disruptions [246].

Therefore, a tailored and adaptive approach is required. As emphasized in this thesis, effective emergency response in multi-robot teams must be underpinned by a **flexible framework** that can be rapidly customized to specific scenarios and system configurations [166]. *Rather than a fixed rule-book, such a framework serves as a toolkit*, enabling system designers and the robots themselves to generate context-aware and coordinated responses. This aligns with the second pillar of this thesis, which explores how cooperation and coordination in multi-robot teams can form the foundation for novel and efficient obstacle and crisis management strategies. Ultimately, the goal is to ensure resilient and intelligent collective behavior, even under unforeseen emergency conditions.

The common approach of avoiding all obstacles inherently involves a basic trade-off; the cost of avoiding obstacles in terms of time and energy against the possible damage of collisions. This thesis formalizes the trade-off and evaluates conditions under which it can improve efficiency. By assessing obstacles not simply as threats but as obstacles with varying damage levels, robots can make strategic choices to avoid or traverse them. *Enhancing overall mission success as outlined in the thesis's decision-making framework [180].*

1.2 Problem Statement

A primary challenge in autonomous robotics lies in optimizing navigation strategies to simultaneously ensure safety, efficiency, and operational reliability. While many path planning methodologies currently prioritize obstacle avoidance, this can result in longer paths and higher energy use in cluttered settings [23, 146]. This uniform avoidance strategy fails to distinguish between obstacles that pose genuine threats and those that incur negligible damage, leading to frequent and often unnecessary avoidance. Consequently, robots may experience significant increases in total path length and energy consumption, potentially compromising their ability to complete missions within critical constraints [127].

The core limitation of existing systems is their lack of a distinct, real-time risk assessment capability. Without the ability to dynamically evaluate the severity of potential collisions, robots cannot make context-sensitive navigation decisions. This results in a suboptimal balance between safety and performance. A decision-making model that intelligently assesses obstacle risks is therefore essential to advance the state of the art [177]. Such a model would enable robots to make informed choices: whether to avoid or traverse, based on a decision-based optimization that weighs potential damage against the costs of deviation, namely increased time and energy outlay.

This problem is further compounded in multi-robot systems, where inefficient individual paths can disrupt overall team coordination and mission efficacy. As illustrated in Fig. 1.2, the cost of avoidance is not merely additive but can become multiplicative in collaborative settings. In crowded or confined environments, such as agricultural fields or warehouse rows, traditional avoidance maneuvers are often infeasible or prohibitively costly. The problem, therefore, extends beyond single-agent navigation to encompass the

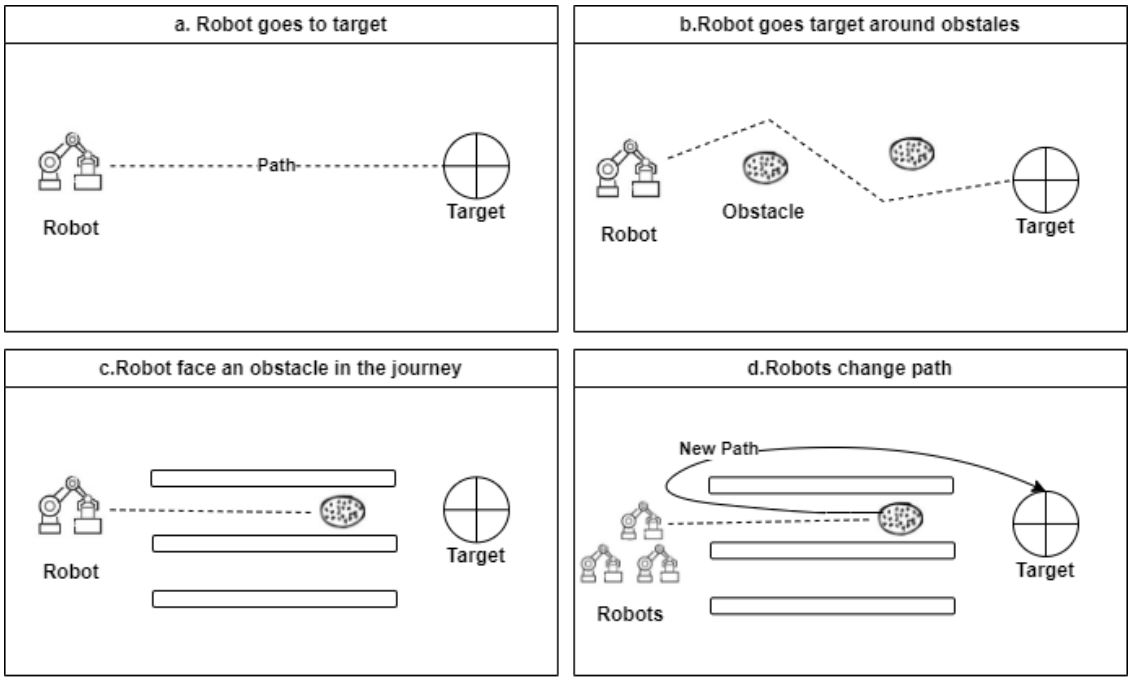


Figure 1.2: Cost of avoiding obstacles.

need for coordinated strategies that maintain system-wide efficiency. *To better clarify the concept and articulate the issue, the following analogy demonstrates the significant cost of avoidance of obstacles in robotic navigation.*

For instance, the robot must travel farther and consume more energy when a path is re-planned. As shown in **panel a** of Fig. 1.2, the most direct path leads straight to the goal. However, in **panel b**, obstacles require a longer path. This becomes more problematic in crowded environments or when multiple robots are working simultaneously.

Panel c shows a different setting, such as an agricultural field where robots move along narrow paths to reach each plant. In these situations, turning around is often not feasible, and avoiding obstacles becomes more costly. In **panel d**, a longer path caused by an obstacle leads to a greater loss of time and energy. Where there is more than one robot, this loss would increase.

In practice, an effective solution must integrate several capabilities:

- **Real-time obstacle characterization** using sensors to estimate properties such as size, material, and mobility [214, 117, 25].
- **Continuous self-assessment** of the robot's own state, including its power levels and structural tolerance.
- **A learning-based predictive model** to estimate the damage cost $D(o_i)$ associated with a collision with an obstacle o_i.
- **A decision function** f compares the estimated damage cost against the predicted path deviation cost $\Delta J = J(\mathcal{P}_{\text{new}}) - J(\mathcal{P})$ to choose the optimal action $\mathcal{A} = \{\text{avoid}, \text{traverse}\}$.

A system that explicitly weighs avoidance costs against contact risk improves robustness in changing environments while preserving safety and efficiency.

Problem Statement

The dissertation addresses path planning for mobile robots in complex environments. Existing approaches commonly treat all obstacles as hard constraints, which can lead to unnecessary detours and excessive energy and time costs. We formulate the problem as a MO task, where the planner must balance a cost function composed of normalized objectives:

$$J(\mathcal{P}) = \alpha \cdot \frac{T(\mathcal{P})}{T_{\max}} + \beta \cdot \frac{E(\mathcal{P})}{E_{\max}} + \gamma \cdot \frac{D(o_i)}{D_{\max}},$$

where $T(\mathcal{P})$ is the travel time, $E(\mathcal{P})$ is the energy consumption, and $D(o_i)$ is the predicted damage cost of contacting the obstacle o_i. The terms $T_{\max}$, $E_{\max}$, and $D_{\max}$ represent plausible upper bounds for each objective, ensuring all terms are dimensionless and comparable.

The core research question is: *How can a UGV decide, for each perceived obstacle, whether to avoid or traverse, by explicitly comparing deviation costs with estimated damage?* The proposed solution integrates a learning model for $D(o_i)$ with a real-time decision function that selects the optimal action $\mathcal{A} \in \{\text{avoid}, \text{traverse}\}$ [173, 181, 177].

1.3 Research Objectives

We, therefore, adopt a two-layer view: we first formulate navigation as an MO problem and then apply MCDM-compatible decision rules, ensuring that preferences never precede the optimization model [178, 173]. Within this formulation, the Multi-objective (MO) stage produces a set of feasible solutions that balance competing objectives, such as safety, energy consumption, and progress toward the goal. Since multiple objectives must be evaluated simultaneously, the subsequent use of Multi-Criteria Decision Making (MCDM) provides a structured way to compare these alternatives. MCDM aggregates the relevant criteria into an interpretable ranking, ensuring that the selection step respects both the optimization results and the relative importance of mission constraints. Building on this principle, the dissertation addresses the fundamental challenges outlined in the thesis formulation. The research objectives comprise a primary focus on risk-aware path planning with explicit damage–efficiency trade-offs and a secondary focus on multi-robot coordination and emergency response.

1.3.1 Primary Objective: Path Planning with Intelligent Risk Assessment

The primary objective is to develop a novel decision-making framework that enables robots to intelligently navigate by balancing path efficiency with potential collision damage. This objective is subdivided into three specific aims:

1. **Formalize the Optimization Problem:** To define a comprehensive cost function $J(\mathcal{P})$ that simultaneously minimizes travel time, energy consumption, and estimated collision damage. This model

will provide the mathematical foundation for trading off the cost of path deviation against the risk of traversing an obstacle.

2. **Develop a Learning-Based Damage Estimation Model:** To implement a machine learning model that dynamically predicts the damage cost $D(o_i)$ associated with colliding with an obstacle o_i. The performance of this model will be quantitatively evaluated based on its accuracy, precision, and recall in classifying obstacle severity.

3. **Design and Integrate a Real-Time Decision Module:** To create a unified decision function $f : (\Delta J,\ D(o_i)) \mapsto \mathcal{A}$ that, in real-time, chooses the optimal action $\mathcal{A}$ (avoid, traverse) by comparing the estimated damage $D(o_i)$ against the path deviation cost $\Delta J = J(\mathcal{P}_{\text{new}}) - J(\mathcal{P})$. The overall system effectiveness will be evaluated in terms of travel time, energy consumption, and collision incidence.

1.3.2 Secondary Objective: A Flexible Framework for Multi-Robot Coordination

The secondary objective is to propose a scalable and adaptable framework for emergency response and coordination in multi-robot teams.

1. **Design a Coordination Protocol:** To develop a framework that enables a team of robots to generate dynamic and coordinated responses to emergencies and obstacles, ensuring robust operation and mission continuity. The framework's effectiveness will be evaluated based on its adaptability, robustness in simulated emergency scenarios, and its ability to minimize systemic inefficiencies.

By fulfilling these objectives, this thesis aims to provide a significant contribution to the fields of autonomous navigation and multi-robot systems by introducing a theoretically grounded and practically applicable framework for intelligent, risk-aware decision-making.

Damage-aware navigation in this dissertation refers to decision-making processes in which the robot evaluates actual or estimated physical damage on its body or components and integrates this damage into the planning or control loops.
Risk-aware navigation, on the other hand, accounts for the probabilistic likelihood of future damage by modeling uncertainty in the environment or in the robot–terrain interaction.
In this context, damage-aware methods provide direct measurements or estimations of incurred damage, while risk-aware methods handle uncertainty by predicting the expected damage. *Thus, damage-aware and risk-aware strategies are not contradictory; risk-aware navigation constitutes a probabilistic extension of damage-aware reasoning.*

1.4 Scope and Significance

This section outlines the boundaries of the research scope and highlights its anticipated contributions to the significance of the field of autonomous robotics.

1.4.1 Scope of the Study

This thesis focuses on algorithmic and methodological advances in risk-aware navigation and multi-robot coordination for autonomous systems. The scope is explicitly defined as follows:

- **Core Focus:** The development and validation of a decision-making framework that integrates path planning with machine learning-based damage estimation for single and multi-robot systems.
- **Included Elements:**
  - Formal mathematical modeling of the path planning problem with damage trade-offs.
  - Design and training of learning models for obstacle risk assessment.
  - Development of anticipatory protocols for multi-robot coordination in obstacle-rich environments.
  - Validation through comprehensive simulation studies representing realistic scenarios in domains such as precision agriculture and warehouse logistics.
- **Excluded Elements:**
  - The design and development of robot-specific hardware or sensor systems.
  - Low-level control algorithms for robot actuators and drives.
  - The logistical challenges of real-world deployment and large-scale hardware integration.
- **Environmental Context:** The research investigates robotic operations in both structured and semi-structured environments, excluding highly unstructured terrains such as open wilderness or underwater applications.

By maintaining this scope, the research provides universally applicable algorithmic insights that can be implemented on various robotic platforms without being constrained by specific hardware configurations.

Scope of This Dissertation

This dissertation investigates path planning for autonomous ground robots, focusing on risk-aware navigation. The problem is framed as balancing classical efficiency measures, such as time or distance, with the potential damage costs of obstacle contact. This extends the traditional obstacle-avoidance paradigm into a learning-based framework, where predicted damage estimates inform decisions to avoid or traverse obstacles. In this way, the quality of navigation improves through the explicit integration of learning and planning.
A second research pillar addresses emergency handling in multi-robot systems. Here, the emphasis is on designing a flexible coordination framework that adapts to diverse scenarios and ensures resilience under disruptions. The framework leverages learning and anticipatory decision rules to enable cooperative responses to obstacles and emergencies. The central hypothesis is that coordinated teamwork provides measurable advantages in safety and efficiency compared to independently operating robots. Together, these two contributions form a coherent investigation into autonomous anticipatory systems, validated through formal models, algorithms, and simulation studies.

1.4.2 Significance of the Research

The contributions of this work are of substantial importance to both theoretical knowledge and practical applications in autonomous robotics:

- **Theoretical Significance:**
  - Introduces a novel paradigm for robotic navigation that moves beyond the conventional *avoid-all-obstacles* doctrine to a nuanced risk-based decision-making approach.
  - Provides a formal risk-aware optimization framework that explicitly models the trade-off between path efficiency and potential collision damage.
  - Advances the theory of anticipatory for multi-robot systems by developing protocols for coordinated emergency response and obstacle management.
- **Practical Significance:**
  - Enables significant improvements in operational efficiency for autonomous robots by reducing unnecessary detours and conserving energy resources.
  - Enhances system resilience through adaptable emergency response strategies tailored for diverse operational environments.
  - Offers implementable solutions for industry applications in logistics, agriculture, and disaster response, where time and energy constraints are critical.
  - Provides a foundation for developing more autonomous, context-aware robotic systems that can operate safely in complex human environments.

The findings and methodologies presented in this thesis are expected to guide future academic research and practical innovations in intelligent robotic systems, contributing to the evolution of more capable and efficient autonomous technologies.

1.5 Notations and Definitions

This section summarizes the notation and key definitions used throughout the thesis to ensure consistency between formulations.

1.1 Environment: The operational space is denoted by $\mathcal{E} \subset \mathbb{R}^n$, where n is the dimension of the workspace (typically $n = 2$ in this work). The environment consists of free space $\mathcal{E}_{free}$ and obstacle space $\mathcal{E}_{obs}$, such that:

$$\mathcal{E}_{free} = \mathcal{E} \setminus \mathcal{E}_{obs}.$$

1.2 Obstacles: Let $\mathcal{O} = \{o_1, o_2, \ldots, o_k\}$ be the set of obstacles in the environment. Each obstacle $o_i \in \mathcal{O}$ is characterized by properties such as position, size, shape, hardness, and mobility.

1.3 Path: A path is indicated by $\mathcal{P}\colon [0,1] \to \mathcal{E}$, representing a continuous mapping from a normalized parameter t to positions in the environment, with:

$$\mathcal{P}(0) = s, \quad \mathcal{P}(1) = g$$

where $s \in \mathcal{E}$ is the starting point and $g \in \mathcal{E}$ is the goal point. In discrete form, a path can be expressed as $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$.

1.4 Path Planning Problem: In the classical setting, the task is to find an optimal path $\mathcal{P}^*$ that is fully contained in free space:

$$\mathcal{P}^* = \arg \min_{\mathcal{P} \subset \mathcal{E}_{free}} J(\mathcal{P}),$$

where $J(\cdot)$ may represent length, traversal time, or energy. In the formulation adopted in this thesis, obstacles are not treated solely as hard constraints: the planner compares the deviation cost ΔJ of avoiding with the estimated damage cost $D(o_i)$ of traversing. The optimal action is chosen by trading off these two terms.

1.5 Damage Estimation Model: A function $D(o_i)$ that returns a predicted numerical cost associated with colliding with an obstacle o_i, based on learned data. The model outputs a raw damage score $D_{\text{raw}}(o_i) \in [0, 100]$. For aggregation and reporting we use the normalized index

$$D(o_i) \;=\; \frac{D_{\text{raw}}(o_i)}{100} \in [0,1],$$

so the objective becomes

$$J(\mathcal{P}) = \alpha \frac{T(\mathcal{P})}{T_{\max}} + \beta \frac{E(\mathcal{P})}{E_{\max}} + \gamma\, D(o_i).$$

Unless stated otherwise, all "Damage Score" values reported in tables/figures are the normalized index $D \in [0,1]$.

1.6 Emergency Scenario: Defined as a tuple $e_i = (s_i, r_i, p_i)$ where:

- s_i – severity level (complexity of environment),
- r_i – required response time,
- p_i – physical or logical components affected.

1.7 Decision Function: A mapping

$$f : (\Delta J,\ D(o_i)) \mapsto \mathcal{A},$$

where $\mathcal{A} = \{\text{avoid}, \text{traverse}\}$. Here $\Delta J = J(\mathcal{P}_{\text{new}}) - J(\mathcal{P})$ denotes the additional path cost of the detour, and $D(o_i)$ is the learned estimate of damage if obstacle o_i is traversed. The decision rule selects the action that minimizes the expected mission cost under the multi-objective formulation.

1.8 Configuration Space (C-Space): The set of all possible robot configurations, denoted by $\mathcal{C} \subset \mathbb{R}^n$, with free and obstacle regions:

$$\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}.$$

1.9 Graph Representation: In discrete planning, the environment is represented as a graph $G = (V, E)$, where V is the set of vertices (states) and E is the set of edges.

1.6 List of Abbreviations

MCDM Multi-Criteria Decision Making

MO Multi-Objective

MOPP Multi-Objective Path Planning

MLR Multiple Linear Regression

ML Machine Learning

DL Deep Learning

RL Reinforcement Learning

SLAM Simultaneous Localization and Mapping

UAV Unmanned Aerial Vehicle

UGV Unmanned Ground Vehicle

CNN Convolutional Neural Network

MDP Markov Decision Process

CPP Coverage Path Planning

MTPP Multi-Target Path Planning

BT Behavior Trees

HC Hierarchical Control

SAP Situation-Aware Planning

PCA Principal Component Analysis

PSO Particle Swarm Optimization

RRT Rapidly-Exploring Random Tree

PRM Probabilistic Roadmap

A* A-star Algorithm

GAs Genetic Algorithms

FLs Fuzzy Logic-Based Algorithms

APF Artificial Potential Field

Chapter 2

Literature Review

2.1 Introduction

In this chapter, we review recent work on path planning and obstacle handling, with an emphasis on adaptive navigation, damage estimation, and emergency response. We summarize the trends and gaps that motivate this thesis—particularly the need to integrate damage estimation with adaptive decision making in dynamic environments—and then describe our search methodology.

2.2 Search Methodology

We surveyed recent work in autonomous robotics following systematic literature review guidelines [153, 94, 124] with a focus on obstacle avoidance, path planning, damage estimation, emergency frameworks, and reinforcement learning. Searches were repeated at several points during the program to capture updates and fill specific gaps.

The following sources were used:

- **Google Scholar:** Chosen for its broad coverage of publication types, including preprints and conference papers. It was used for exploratory queries and to identify recent studies that might not yet appear in indexed journals.
- **Publisher repositories (IEEE Xplore, SpringerLink, ScienceDirect)** are used to complement Google Scholar with peer-reviewed content and stable metadata, thereby reducing the limitations of general-purpose searches.
- **Scopus:** Consulted for curated indexing and citation analysis to identify influential works and research trends.

The procedure was:

Step 1: Access the above databases.

Step 2: Use keyword combinations such as *path planning*, *mobile robots*, *obstacle avoidance*, *multi-agent systems*, *robot damage estimation*, and *emergency response in robotics*.

Step 3: Initial queries typically yielded **50,000–90,000** results (varying over time). We narrowed these down using:

- Relevance to the thesis objectives,
- Journal/conference quality,
- AGH accepted venues,
- Recency (primarily the last five years),
- Technical contribution or algorithmic use.

Step 4: After filtering, approximately 50–100 references were retained for background, comparison, method development, or validation. DOIs were added where available; entries were checked for accuracy and availability.

Step 5: Iterate earlier steps when better or newer studies appear. Duplicates and superseded items were removed. *Only works directly contributing to problem formulation, algorithm design, or evaluation were retained.* Monthly summaries of updates were shared with the supervisor, and citations were placed in the relevant chapters.

2.3 Obstacle Classification, Detection and Avoidance

Accurate identification and categorization of obstacles are prerequisites for safe and efficient navigation. This section synthesizes common taxonomic dimensions used in the literature (e.g., physical nature, visibility, mobility, and traversability) and relates them to planning challenges. A concise summary appears in Table 2.1.

2.3.1 Obstacle Taxonomy

Hard Obstacles

Hard obstacles are rigid physical objects that do not deform or completely block the path of a robot. These must be avoided or navigated to prevent collisions [153]. Common examples include:

1. **Walls:** Fixed vertical barriers that cannot be crossed.
2. **Pillars:** Structural columns that are stationary and obstruct movement.
3. **Rocks/Boulders:** Large natural or artificial formations that cannot be moved.
4. **Buildings/Structures:** Permanent constructions such as houses or industrial facilities.
5. **Vehicles:** Parked or inactive vehicles that block the navigation space.

Table 2.1: Representative obstacle categories in robotic environments [139, 94]

Obstacle Name	**Explanations with Examples**
Traversable	Obstacles that robots can move through, over, or across without significant difficulty. *Examples:* Ramps, stairs, narrow gaps, uneven surfaces, shallow water.
Untraversable	Obstacles that completely block movement and cannot be passed without rerouting. *Examples:* Cliffs, deep pits, large bodies of water, fenced areas.
Visible	Obstacles that can be easily detected by a robot's sensors. *Examples:* Walls, furniture, parked vehicles, trees, poles.
Hidden	Obstacles that are not easily visible but still obstruct movement. *Examples:* Underground pipes, low-hanging objects, potholes, submerged debris.
Soft	Obstacles that are deformable and may yield under contact. *Examples:* Curtains, flexible tubing, vegetation, foam barriers.
Hard	Rigid and immovable obstacles that block the robot's path. *Examples:* Concrete walls, rocks, steel structures, stationary vehicles.
Static	Obstacles that remain fixed in position over time. *Examples:* Buildings, fences, furniture, terrain features.
Dynamic	Obstacles that move or change position, requiring real-time path adjustment. *Examples:* People, animals, moving vehicles, machinery.
Deformable	Objects that change shape when force is applied, affecting how they interact with robots. *Examples:* Inflatable barriers, fabric materials, soft packaging.

Soft Obstacles

Soft obstacles are characterized by their flexible or deformable nature. These objects can change shape, move easily, or change position, making them more difficult to predict and navigate [139]. Examples include both physical materials and non-rigid environmental elements.

1. **Curtains:** Fabric barriers that can bend or move, potentially obstructing the path of a robot.
2. **Flexible Tubes:** Hoses or pipes that can twist or collapse, creating a navigation challenge.
3. **Inflatable Structures:** Objects that expand or contract, changing their shape and affecting mobility.
4. **Soft Vegetation:** Shrubs, grass, or low plants that can bend or sway, partially blocking movement.
5. **Loose Debris:** Materials like sand, gravel, or rubble that can shift and impact traction or movement.

The distinction between hard and soft obstacles is well recognized in robotics, as each type affects path planning in different ways. Studies such as [155] and [242] have analyzed the challenges that soft, flexible obstacles present, particularly in environments that are constantly changing.

Fixed Obstacles

Fixed obstacles are stationary elements in the environment that do not change position. These can include natural and man-made structures such as trees, buildings, poles, or other permanent installations. Their presence requires careful planning to ensure collision-free navigation [139].

Moving Obstacles

Moving obstacles are dynamic elements that change position over time. Examples include people, animals, and vehicles. Since their movements are often unpredictable, navigating around them requires real-time sensing and adaptive path planning [94].

Modern robotic systems typically account for both static and dynamic obstacles. As discussed in studies such as [240] and [65], developing navigation strategies that can handle both types is essential to ensure safety and reliability in dynamic environments.

Visible Obstacles

Visible obstacles are those that can be detected directly by a robot perception system, such as cameras, lidar, or other sensors. These are typically located within the robot's line of sight and are easier to identify and track [153]. The visible obstacles:

- Clearly detectable barriers that block direct movement.
- Upright structural elements that are easily seen and obstruct movement.
- Large, visible natural formations that impede robot mobility.

- Permanent constructions that are visible and require careful path planning.
- Stationary or slow-moving objects that are easily detected, but still pose risks of collision.

Hidden Obstacles

Hidden obstacles are not immediately visible to the sensors and may be partially or completely obscured. These can pose significant challenges for navigation, especially if they are undetected until contact occurs [139].

1. **Underground Pipes:** Buried infrastructure that cannot be detected by surface-level sensors.
2. **Concealed Wires:** Electrical lines hidden inside walls or under floors.
3. **Low-Hanging Objects:** Overhead hazards that may fall outside the typical sensor range.
4. **Hidden Potholes:** Depressions in the terrain covered with foliage or debris.
5. **Submerged Obstacles:** Underwater barriers, such as logs or rocks, that are invisible from the surface.

Research such as [159] highlights the role of sensor fusion and advanced perception algorithms in the effective detection of visible obstacles. Meanwhile, the challenge of identifying hidden obstacles has led to studies on underground and subsurface mapping [203].

Static Obstacles

Static obstacles are fixed objects that do not change position over time. These include structural and environmental elements such as walls, furniture, and buildings. The literature on static obstacles typically focuses on detection, mapping, and planning techniques that allow robots to navigate around them effectively [94].

1. **Walls:** Permanent barriers that block movement and must be avoided.
2. **Pillars:** Vertical supports that obstruct robot paths and require careful navigation.
3. **Furniture:** Fixed items such as tables or desks that make up the navigation environment.
4. **Rocks/Boulders:** Immovable natural features that can obstruct mobile robots.
5. **Buildings/Structures:** Large, permanent constructions that define and limit available paths.

Dynamic Obstacles

Dynamic obstacles are objects that move or change location over time, introducing unpredictability in navigation. These include moving people, vehicles, animals, or machinery. Research in this area focuses on detection, tracking, and predictive modeling to avoid collisions and ensure efficient movement [153].

1. **Moving Vehicles:** Cars, bicycles, or other mobile transport systems that can suddenly change direction or speed.

2. **Pedestrians:** Individuals walking or running who may cross the robot's path unexpectedly.
3. **Animals:** Creatures like dogs, birds, or wildlife that move unpredictably.
4. **Moving Machinery:** Industrial or service equipment that moves location during operation.
5. **Dynamic Structures:** Temporary or movable objects such as scaffolding or construction equipment.

Deformable Obstacles

Deformable obstacles are objects whose shape or size can change in response to external forces. These include soft or flexible materials that may not behave like traditional rigid barriers. Understanding and modeling such obstacles is essential to minimize disruption and prevent damage during navigation [139].

1. **Curtains:** Fabric barriers that can bend and change, temporarily blocking paths.
2. **Flexible Tubes:** Bending or collapsing pipes that complicate traversal.
3. **Inflatable Structures:** Objects that expand or deflate, altering their form.
4. **Soft Vegetation:** Plants or shrubs that can bend with wind or contact.
5. **Loose Debris:** Sand, gravel, or rubble that shifts under movement and affects traction.

Traversable Obstacles

Traversable obstacles are objects that robots can move over or through without significant interference. These include features such as ramps, stairs, slopes, and gaps. Research in this area focuses on identifying such terrains and developing planning methods that allow robots to navigate them efficiently and safely [94].

1. **Ramps and Inclines:** Sloped surfaces that allow vertical movement. Studies address slope path planning by considering factors such as gradient, traction, and robot stability.
2. **Stairs and Steps:** These pose a challenge due to uneven heights and discontinuities. Research explores detection methods, step negotiation algorithms, and adaptive locomotion for safe traversal.
3. **Gaps and Narrow Passages:** Tight spaces that require precise maneuvering. The literature focuses on motion control and obstacle avoidance to pass through narrow areas without collision.
4. **Uneven Terrain:** Surfaces with irregularities such as gravel, vegetation, or rocks. Research emphasizes terrain classification and control strategies to maintain balance and traction.
5. **Water and Other Fluids:** Traversing shallow water or similar environments introduces challenges such as drag and hardware safety. Studies explore waterproofing, buoyancy control, and specialized locomotion methods.

Untraversable Obstacles

Untraversable obstacles are barriers that robots cannot cross directly. These require alternative navigation strategies such as avoidance, rerouting, or high-level decision making. Typical examples include cliffs, deep pits, and large bodies of water that exceed the robot's capabilities [94].

1. **Cliffs:** Steep drops that are not safe to descend or climb.
2. **Deep Pits:** Large depressions in the terrain that robots cannot traverse.
3. **Bodies of Water:** Rivers, lakes, or ponds that require specialized aquatic capabilities.
4. **Impassable Terrain:** Environments like swamps or dense forests where movement is not feasible.
5. **High-Walled Enclosures:** Structures with barriers too high for robots to scale or traverse.

> A structured taxonomy clarifies which obstacle attributes like hardness, visibility, mobility, drive perception, and planning choices. Aligning taxonomy with planning requirements helps map sensing/algorithmic assumptions to operating conditions in a transparent way.

Since different types of obstacles impose distinct sensor and planning requirements, classifying them informs algorithm selection and system design. By tailoring approaches based on obstacle characteristics, robotics researchers can improve perception, decision making, and adaptability in diverse scenarios. Interdisciplinary insights from perception, control, and interaction research play a central role in advancing autonomous capabilities in mobile robots. A comprehensive taxonomy of obstacles is depicted as a branching structure in Fig. 2.1 to enhance clarity and understanding.

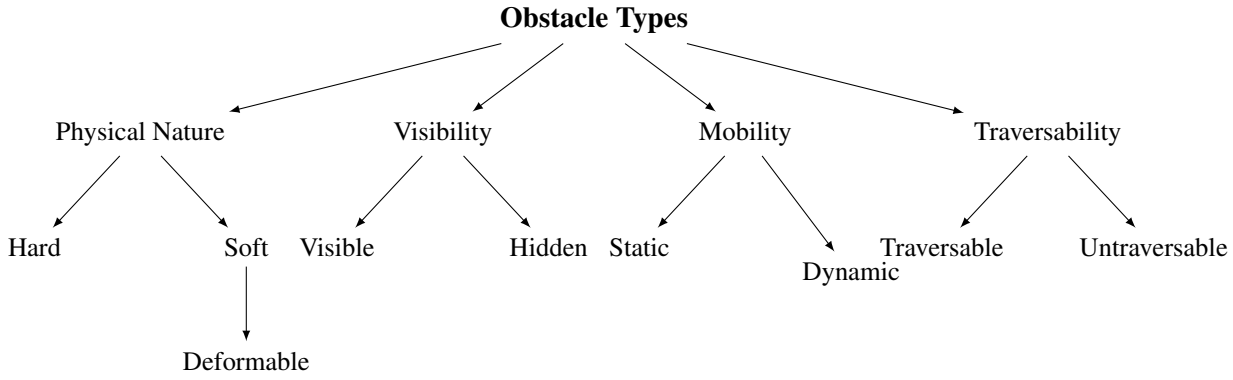


Figure 2.1: Obstacle taxonomy based on physical, perceptual, and navigational attributes.

2.3.2 Obstacle Detection Techniques

This subsection reviews sensing modalities and algorithms used for obstacle detection under real-world conditions (varying weather, illumination, and obstacle density). We consider systems using LiDAR, cameras, radar, ultrasonic sensors, and their fusion.

LiDAR provides high-resolution 3D point clouds with centimeter-level spatial accuracy, enabling precise detection and localization [221, 117, 214]. Limitations include cost, field-of-view constraints, and degraded performance in adverse weather [221].

Camera-based systems are cost-effective and provide rich semantic information for classification and scene understanding. Their performance depends on illumination and can be affected by occlusions and limited depth perception at longer ranges [232].

Radar is robust in fog, rain, and nighttime operation and supports long-range sensing, but has lower spatial resolution than LiDAR or cameras, which can hinder discrimination between nearby objects [130, 118].

Ultrasonic sensors are useful for short-range detection in indoor or confined settings, but their range and resolution limit applicability in high-speed or complex outdoor scenarios [224].

Fusion-based approaches integrate complementary modalities to improve reliability and reduce uncertainty [71, 2]. Typical pipelines combine geometric detail from LiDAR with semantic cues from cameras and kinematic priors from inertial sensors [25]. Fusion increases computational load and requires accurate time synchronization.

Perception algorithms span classical signal processing and modern learning. In vision, object detection, semantic segmentation [41], and optical flow [205] are widely used. LiDAR-based methods operate on point clouds for object and structure extraction [221]. State estimation commonly employs Kalman and particle filters for filtering and multi-sensor integration [161].

Machine learning methods—including convolutional and recurrent architectures—are applied to recognition and scene understanding tasks in robotics [97]. These methods extend perception capabilities in unstructured or visually complex environments, while introducing data and compute requirements that must be considered in real-time systems.

Benchmark Datasets in Obstacle Detection

Benchmark datasets are central to fair evaluation [228]. Widely used resources include KITTI and nuScenes [16], as well as TartanAir for simulated variability; ROS/Gazebo environments enable controlled multi-agent tests.

Once the environment is reconstructed, obstacles must be prioritized relative to mission constraints so that navigation decisions remain consistent with task objectives. The next subsection reviews avoidance strategies.

2.3.3 Obstacle Avoidance

This subsection examines methodologies for obstacle avoidance in mobile robotics, addressing the integrated pipeline from perception to decision-making and highlighting both capabilities and persistent challenges.

Robust avoidance begins with reliable perception. Conventional sensors—LiDAR, radar, ultrasonic, and vision systems—are evaluated on metrics such as detection accuracy, range, resolution, and environmental robustness [117]. To overcome individual limitations, sensor fusion strategies combine the geometric pre-

cision of LiDAR with the semantic richness from cameras, enhancing performance in complex scenarios [25, 23, 247].

Decision-making employs both classical and learning-based algorithms. Established methods like Dijkstra and A* offer computational efficiency and optimality guarantees [119, 234], while machine learning, deep learning [57], and reinforcement learning [236] enable adaptive behavior in evolving contexts, allowing real-time trajectory adjustments [250].

However, significant implementation barriers remain. Learning-based approaches often demand substantial processing power and memory, limiting their use on resource-constrained platforms [156]. Generalization to novel environments is another challenge, with performance drops attributed to overfitting or limited training diversity [168, 126].

Sensor limitations further complicate deployment. LiDAR and cameras struggle in adverse weather, radar lacks spatial detail, and ultrasonics are confined to short-range applications [75, 1]. These constraints can compromise navigation safety in unstructured settings.

As robots increasingly operate alongside humans, social compatibility becomes crucial. Inadequate modeling of social cues and safety margins can undermine trust and collaborative performance [245, 237], making socially aware navigation essential for modern systems.

In summary, while current obstacle avoidance methods demonstrate strengths in sensing and decision-making, practical effectiveness is constrained by computational demands, generalization limits, and human-robot interaction challenges that must be addressed for robust real-world deployment.

2.4 Path Planning Methods and Applications

Understanding obstacle taxonomy is not only crucial for environmental modeling but also directly impacts the selection and design of path planning algorithms. Each obstacle class imposes different computational and physical requirements on the robot's decision-making system. Table 2.2 shows a comparative list of obstacle types and how they affect path planning in different applications.

As shown in Table 2.2, each taxonomy category introduces different planning needs. For example:

- **Dynamic obstacles** require planners to continuously update the path.
- **Traversable obstacles** demand a risk or efficiency trade-off mechanism.
- **Deformable or soft obstacles** often allow for minimal collisions and can be handled using damage-based path adjustment.

Static and hard obstacles, such as walls and buildings, usually require precise global planning algorithms, such as A-star [198]. In contrast, dynamic and deformable obstacles challenge local planning strategies, such as the dynamic window approach or reactive behaviors [114].

Soft or deformable obstacles can be addressed using force field-based methods or reinforcement learning due to their unpredictability [170]. Meanwhile, non-traversable obstacles typically trigger replanning algorithms such as Rapidly Exploring Random Trees or Probabilistic Roadmaps [158, 249].

Table 2.2: Obstacle types and their relevance to planning and applications [116, 250, 9]

Obstacle Type	**Effect on Path Planning**	**Robot Types / Environments**
Static : walls, rocks	Requires initial mapping, can be avoided using global planners	Warehouse robots, indoor delivery systems
Dynamic : humans, animals	Needs reactive/local planners	Autonomous cars, service robots
Deformable : curtains, loose debris	Can sometimes be pushed or ignored; affects collision cost models	Domestic robots, elder-care robotics
Traversable : ramps, grass	May be favored over detours; requires terrain classification	Agriculture, search-and-rescue robots
Untraversable : cliffs, water	Must be strictly avoided; hard constraints in planner	Space exploration, mining
Hidden : potholes, underground cables	Requires sensor fusion; high uncertainty; planners need conservative margins	Field robotics, planetary missions
Soft : vegetation	Often ignored unless damage is expected; influences risk-aware planners	Agricultural swarms, drone missions
Hard : steel walls	Must be strictly avoided; high collision penalty in cost function	Industrial environments, urban navigation

Traversable terrains; such as uneven surfaces and narrow paths, are frequently encountered in agriculture and planetary robotics. In such cases, robots must assess the obstacle type in real time to decide whether to traverse it or avoid [109, 162].

Taxonomy is not an isolated classification task, but a prerequisite step that enables the planner to select the most suitable path planning method, reduce unnecessary detours, and avoid over-conservative or dangerous maneuvers.

2.4.1 Risk-Aware Path Planning

Traditional path planning algorithms in robotics have largely focused on metrics such as the shortest distance or minimum time. Although effective in structured and predictable environments, these approaches often fall short in dynamic or uncertain settings where safety, operational costs, and environmental variability must be considered. To address these limitations, recent research has introduced risk-aware planning frameworks that incorporate uncertainty and potential damage into the planning process [134].

Risk-aware path planning involves assessing and minimizing potential negative outcomes associated with traversing certain areas or interacting with uncertain obstacles. Instead of treating all obstacles equally, these frameworks differentiate between high-risk and low-risk scenarios by considering environmental characteristics, obstacle properties, and the robot's operational context [17].

Several notable algorithms have been proposed in this domain:

- **Chance-Constrained A-star**: Extends the classical A-star by incorporating probabilistic safety constraints that ensure that the likelihood of a collision stays below a predefined threshold [110].
- **Risk-Aware RRT (RA-RRT)**: A variation of Rapidly Exploring Random Trees that biases sampling away from risky regions and dynamically adjusts based on environmental uncertainty [138].
- **Bayesian Decision Networks**: These probabilistic models integrate prior knowledge and sensor data to make risk-sensitive decisions under uncertainty [69].

These methods often require environmental models that include risk maps or cost layers derived from sensor data, historical information, or learning-based estimates. In particular, risk-aware planners often include [59, 217, 246]:

- *Damage estimation cost*: Penalizes paths with high expected physical impact.
- *Energy-risk trade-offs*: Balances efficiency with the likelihood of encounters with obstacles.
- *Real-time adaptability*: Update path based on newly perceived threats or dynamic elements.

Across these studies, multi-objective formulations are frequently used to balance safety, time, energy, and task success under uncertainty. Contemporary planners, therefore, combine probabilistic models, predictive risk assessments, and adaptive cost functions to navigate variable, real-world settings.

2.5 Damage Estimation Models in Robotics

In recent years, learning-based approaches have gained traction as effective tools for estimating potential damage in robotic environments [217]. Unlike physics-based models, which rely on predefined assumptions, machine learning models can extract patterns from large datasets to predict likely outcomes based on real-time sensor input [104, 251]. This adaptability makes them applicable to dynamic and unstructured environments.

However, challenges remain in creating accurate and generalizable damage estimation models. An issue is the lack of large labeled datasets that represent various obstacle interactions. Another issue is the computational overhead of running complex models in real time on resource-constrained robotic platforms [92, 200].

Some models have been developed to estimate damage, ranging from simple heuristic approaches to more complex machine learning (ML) frameworks [46]. These models are widely used in robotics, civil engineering, and disaster management [200], where they help anticipate the potential impacts of external events on systems, structures, and environments. Whether the source is natural, such as earthquakes [92] or floods, or man-made, such as fires and explosions [165], damage estimation methods aim to quantify both scale and severity.

One common approach involves analytical models based on physics and engineering principles. These simulations demonstrate how systems behave under stress, making it possible to predict responses to loads, vibrations, or other external factors [121]. Such models are particularly useful in structural engineering, where they are used to assess the vulnerability of buildings or infrastructure.

Empirical models offer an alternative. Built from historical data, they use statistical techniques to detect patterns and generate predictive insights [83]. These are often applied in disaster planning to forecast the potential effects of hurricanes, floods, or other large-scale emergencies [48].

Recently, the rise of data-driven approaches, especially in ML, has transformed damage estimation. These models leverage large datasets and sensor inputs to capture complex relationships between multiple factors [4]. A key advantage is adaptability. Over time, these models can refine their accuracy as new data become available.

There is also a growing trend toward hybrid models that combine analytical, empirical, and ML techniques [210]. By integrating the strengths of each approach, hybrid models aim for greater robustness and flexibility, especially in environments with high uncertainty or variability.

Damage estimation plays a critical role in enabling systems to adapt, recover, and act safely under unpredictable conditions [56]. This is especially evident in high-risk applications such as search and rescue, where autonomous robots assess collapsed structures or flooded areas to locate survivors [141]. Estimating damage instantaneously helps to guide priorities and allocate emergency resources more effectively.

Damage estimation also supports navigation tasks in agriculture and industrial robotics. For instance, ground robots equipped with multispectral or hyperspectral sensors can evaluate crop health and structural damage caused by pests or extreme weather [51, 31]. Algorithms process environmental data to assess the extent of stress or damage, enabling timely corrective action. Unlike Unmanned Aerial Vehicle (UAV)

platforms, these ground-based solutions directly integrate with terrestrial navigation and are more relevant to the mobile robot frameworks studied in this thesis.

In industrial and hazardous environments, such as nuclear plants or chemical facilities, robots often face conditions that can degrade performance over time: heat, corrosion, or mechanical stress, to name a few [187]. Here, damage estimation models help monitor internal health through sensors that measure temperature, strain, and vibration. These predictive insights improve maintenance planning and can prevent critical failures.

In collaborative environments, especially those involving human-robot interaction, these models are vital for safety [143]. Robots equipped with force or proximity sensors can assess collisions or contact with humans and then react appropriately, braking or stopping motion to prevent injury [92].

Supervised learning methods have been used to estimate the severity of the impact in robotic collision scenarios, as shown in applications such as [225]. These models typically rely on labeled datasets generated from simulations or empirical trials where robots interact with various types of obstacles [129].

2.5.1 A Regressor for Damage Estimation

A regressor aims to learn a functional dependency between a vector of explanatory variables $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$ and a continuous output variable y. In its general form, a regression model approximates the mapping

$$y = f(\mathbf{x}) + \varepsilon, \tag{2.1}$$

where $f(\cdot)$ denotes the underlying predictive function and ε represents the noise from the modeling and measurement. Regression analysis is used to quantify how variations in input factors affect the resulting output and to perform a numerical prediction for previously unseen samples [21].

In this thesis, regressors are used to estimate the continuous *damage score* $y_{\text{damage}} \in [0, 1]$ based on collision-related features such as obstacle hardness, collision velocity, and contact angle. Thus, the problem reduces to learning the mapping

$$y_{\text{damage}} = f(h_{\text{obs}}, v_{\text{col}}, \theta_{\text{col}}), \tag{2.2}$$

allowing the system to assess the expected physical impact of robot–environment interactions.

2.5.2 An Obstacle Classification Problem

The obstacle classification task consists of assigning each obstacle, described by a characteristic vector $\mathbf{x} \in \mathbb{R}^n$, to one of the predefined risk categories K. Formally, the classifier implements a mapping

$$c = g(\mathbf{x}), \qquad c \in \{1, 2, \ldots, K\}, \tag{2.3}$$

where $g(\cdot)$ denotes the decision function, and the class labels correspond to distinct risk levels (e.g., negligible, moderate, critical). The objective is to separate these categories in the feature space such that robot decision-making modules can select an appropriate avoidance or traversal strategy [200].

2.5.3 Neural Networks and CNNs as Supporting Tools

Neural networks (NNs) and convolutional neural networks (CNNs) serve as feature extractors and nonlinear function approximators that can be used jointly with both regression and classification models. Given raw visual or point-cloud measurements $\mathbf{z}$, a neural network computes a representation

$$\mathbf{x} = \phi(\mathbf{z}), \tag{2.4}$$

where $\phi(\cdot)$ denotes the mapping learned by the NN or CNN. The resulting feature vector $\mathbf{x}$ can then be passed to either a regressor of the form (2.1)–(2.2) or to a classifier of the form (2.3), enabling end-to-end estimation of obstacle risk or damage likelihood from high-dimensional sensory data [200].

Recent studies have shown that integrating these models directly into path planning allows for real-time decision making [191]. In these systems, the estimated damage score acts as a dynamic cost function, influencing the planner's decision on whether to avoid or traverse an obstacle. For example, a robot may choose a shorter but riskier path if the predicted damage is within acceptable operational thresholds.

This integration has several benefits:

- Reduce unnecessary detours when the risk of obstacles is minimal.
- Enables proactive rather than reactive behavior.
- Supports multi-objective planning that balances time, energy, and safety.

Damage estimation models help robots adapt their behavior proactively, which is particularly useful in the logistics and industrial inspection domains [56, 187]. Incorporating such models allows robots to make better-informed decisions and respond appropriately to threats, ultimately driving forward the capabilities of autonomous technology in real-world settings.

Ultimately, the development of damage estimation methods reflects a multidisciplinary effort based on engineering, data science, and domain-specific knowledge. A well-rounded model not only predicts risk but also supports timely and informed decisions to reduce harm and improve resilience.

After a careful review, several persistent limitations in damage prediction models are revealed:

- Limited real-world studies combining damage estimation with path planning.
- Few ML-based approaches handle obstacle severity, not just presence.
- Emergency frameworks for robot navigation are underdeveloped in dynamic contexts.

2.6 Decision-Making in Robotics

2.6.1 Formulating the Multi-Criteria Optimization

Let $C \subset \mathbb{R}^n$ denote the configuration space with free space $C_{\text{free}} = C \setminus C_{\text{obs}}$. A kinematically feasible path is $\pi : [0, 1] \to C_{\text{free}}$ with $\pi(0) = q_s$ and $\pi(1) = q_g$. We consider a finite set of mission-relevant performance indices:

$$f(\pi) = \big(f_1(\pi),\, f_2(\pi),\, \ldots,\, f_k(\pi)\big), \tag{2.5}$$

specifically,

$$f_1(\pi) = T(\pi)\ \text{(time)}, \qquad f_2(\pi) = E(\pi)\ \text{(energy)}, \qquad f_3(\pi) = D(\pi)\ \text{(expected damage)}. \tag{2.6}$$

Let $x(t) \in \mathbb{R}^n$ denote the continuous-time state vector of the agent (e.g., position, velocity), and $u(t) \in \mathbb{R}^m$ denote the continuous-time control input vector (e.g., acceleration, steering rate). The system dynamics are governed by the function $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$.

MO Problem: The path-planning task is posed as a vector optimization over the set of admissible continuous-time trajectories:

$$\min_{\pi\in\Pi}\ \mathbf{f}(\pi) \quad \text{subject to} \quad \begin{cases} \pi(t) \in \mathcal{C}_{\text{free}}, & \forall t \in [0,T], \\ \dot{x}(t) = g(x(t),u(t)), & \text{(continuous dynamics)}, \\ u_{\min} \preceq u(t) \preceq u_{\max}, & \text{(control constraints)}, \\ x(t) \in \mathcal{X}_{\text{safe}}, & \text{(state constraints)}, \\ \mathbb{P}\{\text{collision along } \pi\} \le \varepsilon, & \text{(chance constraint)}, \\ \mathcal{E}(\pi) \le \bar{E},\ \mathcal{T}(\pi) \le \bar{T}. & \text{(resource budgets)}. \end{cases} \tag{2.7}$$

Here, Π is the set of admissible paths defined by the dynamics and constraints above. The total trajectory time T may be fixed or free. The objective vector $\mathbf{f}(\pi)$ aggregates competing costs such as path length, smoothness, and risk.

Damage modeling: The damage accumulated along a continuous robot path $\pi : [0,1] \to \mathbb{R}^d$ is quantified by integrating a local hazard field over the arc length of the trajectory. The deterministic formulation is given by

$$D(\pi) = \int_0^1 w^\top \rho(\pi(t))\ \|\dot{\pi}(t)\|\, dt, \tag{2.8}$$

where all symbols are defined as follows:

- $\pi(t)$ — the robot path parameterized over $t \in [0,1]$;
- $\dot{\pi}(t)$ — the time derivative of $\pi(t)$, representing the instantaneous velocity along the path;
- $\rho(q) \in \mathbb{R}^m$ — a vector-valued local hazard field evaluated at configuration q; each component corresponds to a type of instantaneous risk (e.g., structural strain, impact severity);
- $w \in \mathbb{R}^m_{\ge 0}$ — a non-negative weighting vector that scales the contribution of each hazard component;
- $\|\dot{\pi}(t)\|$ — the differential arc-length element ensuring that damage is accumulated per unit distance;
- $D(\pi)$ — the total expected damage along the trajectory.

Under environmental uncertainty ω such as stochastic obstacle properties, the damage is expressed in expectation as

$$\mathbb{E}[D(\pi, \omega)]\,, \tag{2.9}$$

which averages the accumulated damage with respect to the distribution of ω.

On the differentiability of the path. The assumption that π is differentiable is standard in trajectory optimization and path–functional formulations since

$$\|\dot{\pi}(t)\|\, dt \tag{2.10}$$

corresponds to an infinitesimal arc-length element. Any path used by the evaluated algorithms is either constructed as a continuously differentiable curve or can be approximated arbitrarily well by a C^1 curve, ensuring that the integral in (2.8) is well-defined. A similar formulation was adopted in [20].

From MO to a single actionable choice: Once $\Pi^\star$ (or an approximation thereof) is computed, a *decision rule* is applied *after* optimization to select a single path for execution. Two standard, MCDM-compatible selectors are:

- **Weighted (linear) scalarization (used for multi-objective selection):**

$$J_\lambda(\pi) = \sum_{i=1}^{k} \lambda_i\, f_i(\pi), \qquad \lambda_i \geq 0, \quad \sum_{i=1}^{k} \lambda_i = 1. \tag{2.11}$$

  where:

  - π — candidate path;
  - $f_i(\pi)$ — the i-th objective function evaluated on path π, such as path length and damage risk;
  - k — number of objectives;
  - λ_i — non-negative scalarization weights that control the contributions of each objective;
  - $J_\lambda(\pi)$ — single-objective score used for selection.

- **ε-constraint selection:** One objective is minimized while the remaining ones are constrained:

$$\min_{\pi}\ f_j(\pi) \quad \text{subject to} \quad f_i(\pi) \leq \varepsilon_i, \quad i \neq j. \tag{2.12}$$

  where:

  - $f_j(\pi)$ — primary objective selected for minimization;
  - $f_i(\pi)$ — remaining objectives treated as constraints;
  - ε_i — admissible upper bounds for each constrained objective;
  - π — feasible path satisfying all constraints.

Principle

MCDM follows MO: preferences (λ, ε levels, lexicographic orders, etc.) are used *only after* the feasible set and Pareto front of (2.7) are established. Thus, *preferences never precede the optimization model.*

Computational note: In practice, $\Pi^\star$ is approximated using sampling-based planners or grid/graph search, augmented with uncertainty-aware costs or chance constraints. The decision rule then selects one representative path from the approximated front $\widehat{\mathcal{F}}^\star$ for execution.

2.6.2 Multi-Criteria Decision-Making (MCDM) in Robotic Navigation

In environments with limited resources and unpredictable obstacles, autonomous robots must evaluate multiple competing factors when selecting navigation strategies. MCDM frameworks provide structured methods to balance trade-offs among conflicting objectives, such as minimizing energy consumption, reducing travel time, and avoiding collision-induced damage [61].

MCDM methods have been widely studied in other engineering domains but are increasingly being adopted in robotics. These techniques allow decision making to be formulated as an optimization problem that involves several weighted criteria [10]. In the context of obstacle handling, a composite cost is often used:

$$C = \alpha \cdot \frac{T}{T_{\max}} + \beta \cdot \frac{E}{E_{\max}} + \gamma \cdot \frac{D(o_i)}{D_{\max}}, \tag{2.13}$$

where:

- $\alpha, \beta, \gamma \geq 0$ are mission-dependent weights [61]
- T is the estimated time to reach the goal.
- E is the projected energy consumption.
- $D(o)$ is the damage predicted by the interaction with the obstacle o.

Several MCDM techniques are applicable to robotic decision making:

- **Weighted Sum Models (WSM)** combine multiple criteria into a single scalar score, allowing fast comparisons between alternative paths [61].
- **Fuzzy Logic-based MCDM:** It handles vague or imprecise input, which is useful when damage or energy costs cannot be precisely quantified [238].

Integrating MCDM into navigation frameworks requires a reliable flow of information from robot perception systems. Robots rely on sensors such as cameras, lidar, radar, and ultrasonic devices to collect environmental data [117], and sensor fusion techniques combine these inputs into a consistent situational model [2]. This processed perception output is fed directly into the MCDM cost functions, ensuring that decisions are based on accurate and timely environmental understanding.

Planning and control then translate these decisions into executable actions. Path-planning algorithms define feasible routes that meet the weighted objectives, while control algorithms ensure the precise execution of movements [57, 49]. In this way, MCDM does not replace traditional planning and control but enhances them by making their outputs explicitly goal- and context-aware.

In the case of obstacle avoidance, MCDM enables the evaluation of the location, size, motion, and risk level of nearby objects [160, 59, 33]. By combining this information with mission priorities and physical

constraints, robots can select trajectories that balance safety, efficiency, and energy use [184] . Damage estimation models can be integrated into the cost function $D(o)$, allowing risk to be quantitatively weighed against other mission goals.

Despite its advantages, integrating MCDM into real-time systems remains computationally challenging, especially when sensor data are noisy or incomplete [136]. Probabilistic and risk-aware reasoning frameworks [134] further improve the robustness of decisions in uncertain environments.

MCDM provides a structured foundation for balancing multiple mission objectives in robotic navigation. By linking perception, planning, control, and risk assessment through a unified cost function, robots can make informed trade-offs that improve adaptability and safety in complex and uncertain settings.

Learning-based decision strategies, including reinforcement learning and its multicriteria extensions, are discussed in the next chapter, where they are analyzed alongside other machine learning techniques in the context of adaptive navigation.

2.6.3 Cooperative Decision Strategies

Before examining specific implementations, it is useful to outline the principles behind swarm robotics and cooperative decision-making. Drawing inspiration from collective behavior observed in nature, such as foraging in ants or flocking in birds, swarm-based strategies emphasize decentralized coordination, redundancy, and the emergence of complex group behavior from simple local rules [132].

In robotic navigation, cooperative approaches often employ methods such as potential fields, consensus algorithms, and distributed fuzzy control. These techniques enable groups of robots to negotiate obstacle handling, allocate tasks, and adjust movement patterns collectively [64]. Such systems must address challenges, including real-time negotiation, conflict resolution when priorities differ, and maintaining effective operation with minimal communication overhead.

The mechanisms developed in these cooperative settings form a natural foundation for more advanced frameworks. Building on established cooperative principles, Multi-Agent Reinforcement Learning (MARL) extends these capabilities with adaptive learning and long-horizon strategy optimization in multi-robot environments [250].

> Although this thesis does not focus directly on space robotics, planetary rovers offer valuable insight into some of the most advanced autonomous systems ever built. Their operation in extreme and unstructured environments provides a benchmark for what resilient, adaptive, and intelligent robotic autonomy should look like.

2.7 Obstacle Avoidance in Advanced Robotic Systems

This section reviews learning strategies, obstacle handling, and emergency response mechanisms developed for planetary exploration and summarizes which principles transfer to terrestrial robotics in high-risk or communication-constrained scenarios.

The discussion also provides context for this dissertation: several methods originally developed for planetary rovers illustrate how autonomous systems can remain functional under uncertainty and limited supervision.

Navigating Extreme Terrains: Rovers operating on planets and moons must traverse steep slopes, brittle ice, loose sand, and high-pressure atmospheres [72, 66, 113]. Typical mobility solutions include rocker–bogie suspensions, specialized wheels, and traction control. For subsurface access, such as icy moons, missions use thermal drills and impact-resistant tools [212]. These choices reflect constraints on traction, stability, and energy management in sparsely mapped environments [34].

Environmental Challenges: Mission environments introduce radiation, temperature extremes, corrosive atmospheres, and fine dust. On Venus, high temperatures and chemical reactivity dominate; on Mars, dust storms reduce solar yield and affect sensing [189]. Mitigations include sealed housings, thermal control, radiation shielding, and dust management. Power and communication are also limiting: many missions rely on solar arrays, with radioisotope systems used where insolation is insufficient; delayed or intermittent communications are handled via relay assets and increased onboard autonomy [124, 109].

Emergency Responses in Planetary Missions: Table 2.3 summarizes common emergencies and corresponding responses drawn from prior missions and engineering studies.

Table 2.3: Summary of emergency scenarios and robotic responses [212, 210, 135, 68]

Emergency Scenario	**Response Strategy**	**Typical Application**
Steep cliffs or vertical obstacles	Gripping tools; anchor-point detection using sensors and stereo vision	Lunar, Martian terrain
Loose sand or granular surfaces	Adaptive wheels, cleats, traction control systems	Mars (dunes, regolith)
Extreme weather or temperature shifts	Thermal insulation, radiation shielding, robust housing	Mars, Venus, Moon
High radiation exposure	Radiation-hardened components; automatic shutdown procedures	Mars, Europa, Ganymede
Corrosive environments (sulfuric acid)	Corrosion-resistant materials; sealed enclosures	Venus
Subsurface access requirements	Thermal drills; impact-resistant penetration tools	Europa, Enceladus
Low-gravity mobility challenges	Lightweight frames, precise locomotion algorithms	Asteroids, small moons

Case Reflections: Mars and Venus Missions: Mars rovers (Spirit, Opportunity, Curiosity, Perseverance) demonstrate how robust engineering and onboard navigation extend mission lifetimes and scientific return

[140, 103, 6]. The Perseverance mission also integrated an aerial scout (Ingenuity) to augment situational awareness [68]. Earlier Venus missions reported operations under high pressure and temperature, informing designs for dense, acidic atmospheres [44].

Robotic Failures

Documented failures highlight limits of perception and mobility. For example, NASA's Spirit rover became immobilized in soft terrain [6], and the DARPA Robotics Challenge featured falls and stalls linked to sensing and control issues [202]. These cases motivate explicit risk assessment, environment modeling, and system redundancy in mission planning.

2.8 Emergency Planning in Robotic Environments

As mentioned before, much attention has been devoted to reactive obstacle avoidance and real-time navigation; however, relatively few studies have addressed emergency response planning in autonomous robotic systems. Accordingly, structured yet adaptable emergency-response schemes are needed to support operational continuity across domains [38, 56]. *Therefore, creating structured yet adaptable emergency response frameworks is critical to operational continuity in both terrestrial and extraterrestrial applications.*

Traditional emergency handling mechanisms in robotics are often based on rules that are predefined for specific scenarios. Although effective in narrow contexts, these rigid systems fail when exposed to novel, unstructured environments or mission-specific constraints [38]. This has led researchers to propose more flexible and modular approaches that are capable of dynamically adjusting to unforeseen situations.

Several trends in the literature aim to address this challenge:

- **Behavior Trees (BT):** A BT is a rooted directed tree $BT = (V, E)$ where each node $v \in V$, following the formalization in [35], is either a control-flow node (selector, sequence, parallel) or a task node (action, condition). Each node executes a state-transition function

$$\tau(v, t) \in \{\text{Success},\ \text{Failure},\ \text{Running}\}, \tag{2.14}$$

  where t denotes the discrete tick index, namely, the evaluation step at which the node is updated. The return state is propagated recursively from children to their parent, enabling contingency handling by switching branches when failures occur. This makes BTs particularly suitable for emergency behavior design in uncertain environments.

- **Hierarchical Control (HC):** A hierarchical controller consists of a stack of layers $\{C_0, \ldots, C_L\}$, where higher layers handle long-horizon or abstract decisions and lower layers handle fast, local control actions. Each layer C_ℓ solves a constrained optimization problem of the form

$$u_\ell^\star(t) = \arg\min_{u_\ell}\ J_\ell(x_\ell(t), u_\ell) \quad \text{s.t.} \quad g_\ell\big(x_\ell(t), u_\ell\big) \le 0, \tag{2.15}$$

  where:

  – t — discrete control step or time index;

- $\ell \in \{0, \ldots, L\}$ — layer index (with $\ell = 0$ the lowest, fast-reactive layer);
- $x_\ell(t)$ — state vector available at layer C_ℓ at time t;
- u_ℓ — control input considered by layer C_ℓ;
- $u_\ell^\star(t)$ — optimal control action returned by layer C_ℓ;
- $J_\ell(x_\ell, u_\ell)$ — layer-specific cost function (e.g., tracking error, energy cost);
- $g_\ell(x_\ell, u_\ell)$ — constraint functions ensuring feasibility (e.g., safety, actuator limits, and collision margins).

The output of layer C_ℓ (a control command or reference trajectory) is passed downward and becomes a target or constraint for the next layer $C_{\ell-1}$. This ensures recursive feasibility and stability while enabling fallback to higher-level controllers under faults or unexpected disturbances [99].

- **Situation-Aware Planning (SAP):** The decision-making module is expressed as a mapping

$$\Phi : \mathcal{S} \times \mathcal{M} \to \mathcal{A}, \tag{2.16}$$

where:

- $\mathcal{S}$ — set of perceived environmental states, such as obstacles, threats, and robot status;
- $\mathcal{M}$ — mission context, including task objectives, priority levels, and operational constraints (weather, rules of engagement, battery state, etc.);
- $\mathcal{A}$ — set of admissible actions or maneuvers;
- Φ — situation-aware policy that maps the current situation to an action.

At each evaluation step, the selected action maximizes a mission-dependent utility:

$$a^\star = \arg\max_{a \in \mathcal{A}} U(s, m, a), \tag{2.17}$$

where:

- $s \in \mathcal{S}$ — current state observation,
- $m \in \mathcal{M}$ — active mission context,
- $U(s, m, a)$ — utility function encoding mission performance, risk penalties, and safety constraints.

This formulation enables context-dependent adaptation of robot actions while ensuring that mission priorities and safety requirements are consistently enforced [96].

- **Probabilistic Risk Models:** Define a finite set of failure modes $\mathcal{F} = \{f_1, \ldots, f_k\}$ with probabilities $p(f_i)$ and impact costs $c(f_i)$. The expected residual risk is defined as

$$R = \sum_{i=1}^{k} p(f_i)\, c(f_i). \tag{2.18}$$

Since R does not depend on the action, emergency strategies are chosen to minimize the overall residual risk under feasibility constraints, i.e.,

$$a^\star = \arg\min_{a \in \mathcal{A}} R.$$

[246]

- **Policy-Based Emergency Response:** Modeled as a policy function

$$\pi : \mathcal{X} \to \mathcal{A}, \tag{2.19}$$

where $\mathcal{X}$ is the system state space and $\mathcal{A}$ is the action set. Policies can be logic-based (finite-state automata, rule sets) or data-driven (learned classifiers). The emergency action at time t is

$$a_t = \pi(x_t), \tag{2.20}$$

ensuring real-time response by direct state-to-action mapping [171].

In addition, recent research emphasizes the importance of domain-specific adaptability. For example:

- In **agricultural robots**, emergency strategies must handle irregular terrain, sensor occlusion, and sudden biological interference [13].
- In **search and rescue robots**, the ability to continue operation without sensor failure or debris trapping is crucial to saving lives [52].
- In **planetary robots**, where communication is delayed or absent, autonomous fallback strategies must be activated immediately and reliably [190].

Studies repeatedly emphasize that emergency planning should not be something that should be thought about later, but rather an integrated part of navigation and decision making. Adaptive emergency plans must consider robot hardware, environmental characteristics, mission goals, and contextual data to deliver timely, effective, and safe responses.

Recent work highlights the utility of modular emergency-planning frameworks that can be tailored to different robotic environments. Such frameworks support configuration by mission-specific parameters and prioritize continuity while minimizing damage and downtime during critical events.

2.9 Summary

This chapter reviewed methods and challenges in obstacle detection, avoidance, damage estimation, and decision making for autonomous robots. The survey covered sensing modalities, classical and learning-based algorithms, and case studies from terrestrial and planetary robotics.

Recent studies indicate a growing reliance on predictive and learning-based techniques in navigation and decision making. In particular, deep learning and reinforcement learning have been applied to improve robustness in unstructured environments [57, 236]. These methods complement traditional approaches but still face challenges related to computational cost, data requirements, and generalization across unseen conditions.

Research on damage estimation emphasizes that robots must consider not only the presence of obstacles but also the severity of potential interactions [56, 141]. This reflects an increasing focus on safety-aware autonomy, where navigation choices balance task success with system integrity.

Despite progress, the development of fully integrated real-time systems remains difficult. Field deployments and planetary rover missions reveal persistent limitations in perception, adaptation, and resource management under uncertainty [202]. These constraints affect both hardware reliability and control strategies.

A promising direction is the use of hybrid approaches that combine model-based planning with learning-based adaptation. Coupling simulated training with real-world validation has been proposed as a way to bridge the gap between research prototypes and operational systems.

Advances in sensing, planning, and learning have improved robustness, yet systems still face uncertainty and incomplete information. The literature indicates ongoing needs in integrating damage estimation with adaptive decision making for real-time navigation.

Chapter 3

ML-based Advances in Autonomous Robot Systems

This chapter focuses on ML techniques—supervised, unsupervised, deep learning, and reinforcement learning—that directly influence autonomy in robotic systems. Rather than conducting a broad survey, we outline how these methods contribute to perception, decision-making, and planning in real-world applications relevant to this dissertation.

We treat ML in a separate chapter because it is a core component of the thesis. First, we summarize recent advances to position the techniques we use; then, we emphasize optimization strategies that matter for route planning performance. Fig. 3.1 situates this chapter within the overall methodology.

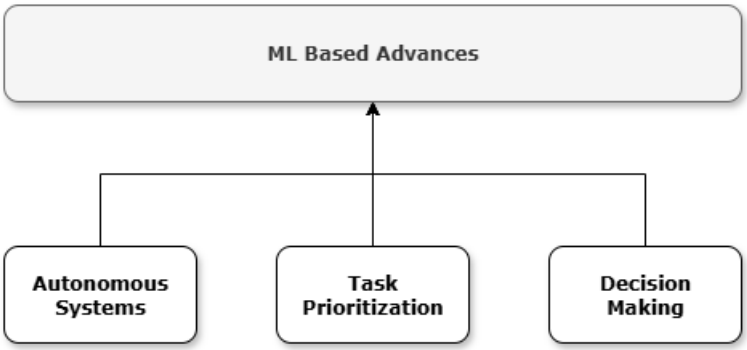


Figure 3.1: Core pillars of this chapter.

The terminology surrounding autonomy varies across subfields. In robotics, the term *autonomous robot* generally denotes a system that perceives, decides, and acts without continuous human supervision; this contrasts with automated or teleoperated platforms [5, 157]. In mobile robotics, usage differs by context, but the core attributes—onboard perception, decision making, and execution—remain consistent. Representative application domains include precision agriculture, manufacturing, and space robotics [162, 160, 19]. Throughout this chapter, "autonomous system" and "mobile robot" are used in this functional sense, with task-specific assumptions stated when necessary.

A detailed methodology diagram combining ML with decision making and risk assessment appears later, where these components are integrated into the planning pipeline.

Unlike rule-based systems, machine learning enables autonomous agents to adapt to unseen scenarios by learning complex policies from data, a key factor in transitioning from reactive control to adaptive behavior [5]. For example, supervised learning stands out for its ability to train systems using labeled datasets, allowing them to recognize input-output relationships and generalize to new cases [208]. On the other hand, unsupervised learning explores data without predefined categories or outcomes. This approach is particularly useful when systems must identify hidden patterns or groupings, fostering a level of adaptability that is often difficult to achieve with traditional methods [241].

Another paradigm is RL, which draws inspiration from behavioral psychology. Here, learning is driven by trial and error, and agents receive feedback in the form of rewards or penalties. Over time, this helps optimize decision-making, particularly in dynamic and uncertain environments [101].

Taken together, these approaches allow autonomous systems to interpret their surroundings, make context-sensitive decisions, and adapt over time. Integrating ML into perception and planning shifts systems from purely reactive control to adaptive behavior grounded in data. Below, we outline the core ML approaches and their roles in autonomy.

3.1 ML in Autonomous Systems

ML refers to algorithms that improve task performance by identifying patterns in data and adjusting model parameters accordingly. Formally, given training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, the goal is to approximate a function f_θ that minimizes a task-specific loss L [172]:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(f_\theta(x_i), y_i). \tag{3.1}$$

Supervised Learning (SL): Algorithms are trained with labeled pairs (x_i, y_i). The objective is to generalize the mapping $x \mapsto y$ to unseen data. Applications in robotics include terrain classification, object detection, and sensor calibration [254].

Unsupervised Learning (UL): Operates on unlabeled data $\{x_i\}_{i=1}^{N}$ to uncover latent structure. In clustering, data are partitioned into K groups by minimizing within-cluster variance using the K-means objective [55]:

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2. \tag{3.2}$$

Robotic use cases include anomaly detection and feature extraction [223].

Deep Learning (DL): A class of ML methods based on neural networks with multiple layers. CNNs are widely applied for visual recognition, while recurrent and attention-based models support sequence prediction and scene understanding [251]. In robotics, DL processes multimodal inputs such as images, LiDAR point clouds, and radar signals.

Reinforcement Learning (RL): RL is usually formalized as a Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma \rangle$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ denotes the action set, $\mathcal{P}$ indicates the transition probabilities, R corresponds to the reward function, and γ refers to the discount factor. The agent seeks a policy $\pi(a|s)$ that maximizes expected return:

$$J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]. \tag{3.3}$$

This framework has been applied to navigation, control, and adaptive decision making under uncertainty [101, 115]. Training uses labeled or unlabeled data to learn representations and policies that generalize beyond the training set [24].

Among the various ML approaches, **supervised learning** stands out for its practical applications in terrain classification and object detection. In this method, robots are trained on labeled datasets, allowing them to recognize environmental characteristics such as rocky surfaces, cliffs, or soft sand based on previous examples [254]. For example, annotated images collected from earlier missions can serve as training material, allowing the robot to distinguish between hazardous and navigable terrain. Similarly, supervised learning models help identify scientific targets, such as ice deposits or mineral-rich formations, thus enhancing mission planning and execution. In scenarios where only a subset of the data can be labeled—due to limited expert availability or communication constraints—semi-supervised learning techniques can be employed, enabling the robot to leverage both labeled and unlabeled samples to improve classification performance with minimal annotation effort.

Reinforcement Learning (RL) in practice: Beyond the formal MDP definition, RL has been applied to robotic navigation and control. For example, a planetary rover can model traversable terrain as states, candidate moves as actions, and define rewards for progress toward a goal while penalizing collisions or excessive energy use. By iteratively updating its policy, the rover converges on routes that balance safety and efficiency under uncertainty [101, 115].

Deep Learning (DL) in robotics: DL architectures process multimodal sensor inputs, enabling robots to integrate vision, LiDAR, and radar for robust perception [251]. CNNs are used for object and terrain classification, while recurrent and attention-based models handle sequential or temporal data. For instance, CNN-based terrain classifiers allow a robot to distinguish between safe ground and obstacles, while fusion of camera and LiDAR features supports real-time obstacle detection. Such models enhance situational awareness but require high computational resources and careful training to generalize reliably.

Importantly, many ML systems are designed to continue learning from the incoming data, which allows them to remain effective under changing conditions. This ability to adapt over time is especially valuable in autonomous systems, where the environment may evolve unpredictably.

3.1.1 Applications

The integration of ML into robotics has enabled more adaptive systems [233]. Modern robots increasingly leverage online learning and sensory feedback to update policies for operation in changing conditions [115], supporting navigation, interaction, and task execution with reduced reliance on manual supervision [226, 12].

Table 3.1: Key challenges and solutions in ML for autonomous systems [211]

Challenge	Solutions
Restricted Data Availability	Data augmentation, transfer learning, active learning for key samples
Real-Time Processing Limitations	Model architecture optimization, GPU/TPU acceleration, distributed/edge computing
Model Interpretability	Transparent models (decision trees), post-hoc tools (SHAP, LIME), user-level explanations
Safety and Reliability	Redundancy and fail-safes, simulation testing, uncertainty estimation
Adaptability to Changing Environments	Continuous learning frameworks, sensor fusion, reinforcement learning

In industrial settings, ML enables predictive maintenance by detecting wear or malfunction based on sensor data and assists in quality control through automated defect detection [50]. In space robotics, intelligent systems equipped with learning algorithms improve autonomous navigation and terrain classification on unfamiliar planetary surfaces [254].

A particularly impactful application has emerged in autonomous vehicles. Learning algorithms are central to interpreting sensor data, planning, and executing safe navigation strategies [219]. Supervised learning enables the detection of lane markings, pedestrians, and other vehicles, while RL supports adaptation to dynamic road conditions by learning from feedback [225]. These systems continuously improve as they encounter new scenarios, which is essential for robustness in safety-critical domains [58].

UAVs as a parallel domain: A significant line of research investigates UAVs, where ML techniques support navigation, obstacle avoidance, and environmental monitoring [11, 229]. UAVs pose distinct aerodynamic and control challenges; yet, many perception and decision-making principles overlap with terrestrial robotics. Since this dissertation focuses on ground-based mobile robots, UAV-related work is only briefly noted here and is not analyzed in detail.

3.1.2 Challenges and Solutions

Key Challenges: The integration of ML into autonomous systems introduces a variety of challenges that extend across technical, practical, and ethical dimensions. One of the greatest difficulties lies in training models to perform reliably in dynamic and often unpredictable environments [207]. Unlike controlled scenarios, real-world conditions may involve unexpected edge cases that are underrepresented in training data, which can hinder model generalization and robustness.

Another critical concern is the reliability and safety of such systems [37]. ML models, particularly those deployed in autonomous agents, can behave unpredictably when exposed to unfamiliar input patterns. This raises important questions about the dependability of systems tasked with making decisions without human intervention.

Another limitation involves data availability. Developing high-quality models requires access to large, diverse, and well-annotated datasets—resources that are not always readily available, especially in specialized fields [145]. The challenge of acquiring representative data makes it difficult to train systems that perform well in all relevant scenarios. Overcoming these hurdles calls for collaboration across domains, combining knowledge from ML, robotics, and the specific application area.

One such issue is **data privacy**, as these systems often depend on large volumes of personal or sensitive information for learning and decision making [244, 197]. For example, in autonomous healthcare systems, mobile rehabilitation or care robots carry sensitive patient data, such as medical histories and biometric indicators, that must be processed for diagnostics or decision making, raising significant concerns about privacy compliance and data governance under regulations.

An additional issue that continues to attract attention is the question of model interpretability. As ML systems become more complex, particularly with the rise of deep learning architectures, their internal decision-making processes often become opaque. These so-called "black box" models can deliver high performance, yet offer little insight into how or why a particular outcome was reached [167]. In contexts where systems influence human safety, legal rights, or ethical outcomes, this lack of transparency poses a serious barrier to accountability and public trust. For example, in the judicial or financial domains where ML models are used to assess credit risk [18] or sentencing recommendations, lack of interpretability can lead to opaque and potentially biased results, making explainability not just a technical goal but a legal and ethical imperative.

Another layer of complexity arises from the demand for **real-time decision making**, especially in domains where timing is critical. In the case of autonomous vehicles, for example, decisions must be made instantly and under uncertainty [120]. For example, in autonomous vehicles, detecting a pedestrian suddenly walking onto the road and applying the brakes within milliseconds requires not only rapid sensor fusion and prediction but also extremely low-latency inference from on-board models [252].

Collectively, these concerns highlight the importance of a thoughtful and interdisciplinary approach in the development of autonomous systems. This includes collaboration between ML experts, roboticists, cognitive scientists, and ethicists to ensure that both technical functionality and societal considerations are addressed in tandem.

In addition, integrating principles from human factors and systems engineering can help design more robust and user-centric autonomous systems. These perspectives contribute to understanding how humans interact with AI and how those interactions can be designed for transparency, safety, and trustworthiness [63].

In practice, this collaborative ecosystem also involves policymakers, legal experts, and standards organizations that define boundaries and regulations for the ethical deployment of intelligent systems.

Proposed Solutions: Addressing the challenges associated with the application of ML in autonomous systems requires continued research and thoughtful innovation. One promising direction is the development of advanced simulation platforms in which autonomous agents can be trained in virtual environments [4]. This not only reduces dependence on real-world data collection, but also mitigates safety risks during the early stages of system development.

Concerns surrounding data privacy can be addressed using techniques such as federated learning or differential privacy, both of which allow sensitive data to remain decentralized during training while still contributing to model improvement. When it comes to improving the transparency of decision-making processes, the growing field of explainable AI offers methods to make complex models more interpretable and understandable [160].

Meanwhile, challenges related to **real-time decision making** have led to increased interest in high-performance computing strategies such as *edge computing*, which reduce latency by enabling computation directly on or near the device [125].

Ultimately, solving these challenges requires cooperation between academic institutions, industry stakeholders, and regulatory bodies. Only through collaborative efforts can ML-based autonomous systems be technically robust and socially responsible. An overview of key challenges and potential solutions is summarized in Table 3.1.

3.2 ML in Task Prioritization

Task prioritization is most commonly modeled as a Multi-Criteria Decision-Making (MCDM) problem, where tasks are ranked or sorted according to urgency, importance, and resource constraints [28]. *MCDM provides structured methodologies for evaluating and selecting among alternatives based on multiple, often conflicting criteria.* Criteria may include mission-criticality, safety, energy usage, or operator-defined weights. Classical MCDM methods or outranking procedures are frequently applied in robotics for sequencing tasks under uncertainty [134, 248].

Empirical or ML-based methods play a secondary role at this stage but can contribute to *preference learning*, where human or contextual preferences are inferred from data and then integrated with MCDM. This hybrid approach yields structured rankings or sorting consistent with decision science principles [231]. In this way, data-driven methods enhance but do not replace the formal decision-analytic foundation of prioritization.

In the context of autonomous robots, effective prioritization ensures that urgent actions, like obstacle avoidance, are executed immediately, while less critical or long-term objectives are scheduled appropriately. This structured allocation improves resource use, maintains safety, and aligns robot actions with mission goals in complex environments.

At its core, task prioritization refers to the organization of actions by relevance and timing. Each task receives a defined priority level that determines its place in the execution queue. This structured approach supports better resource allocation and situational responsiveness [98]. As a result, robots can operate more effectively in dynamic or uncertain environments [2, 201]. To gain an intuitive perspective on the concept of task prioritization, we generated a visualization, as shown in Fig. 3.2.

Looking at traditional prioritization techniques helps us understand the evolution of these systems. The early approaches were primarily rule-based, relying on fixed logic or heuristics provided by human experts [253]. Although these methods are easy to interpret and implement, their rigidity becomes a drawback in dynamic or poorly defined environments [108].

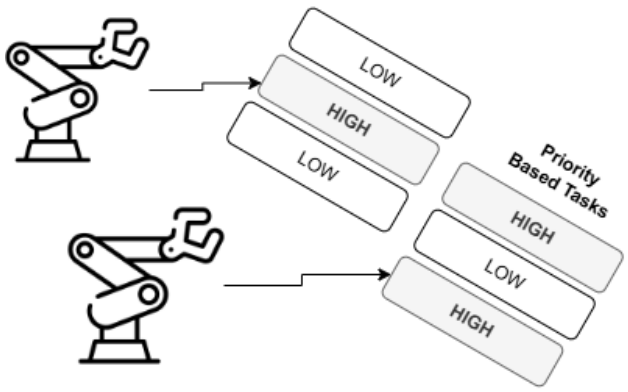


Figure 3.2: A visual representing task prioritization.

3.2.1 Prioritization with ML

While MCDM provides the principal framework for task prioritization, machine learning can support this process in a complementary role. In particular, ML is useful for *preference learning*, where robot behavior is tuned based on empirical observations or operator feedback rather than fixed weights [84]. Learned preferences can then be incorporated into classical MCDM methods to generate rankings or sorting rules that remain consistent with decision-analytic principles.

Another supportive use of ML lies in estimating context-dependent criteria. For example, regression models can approximate energy consumption for candidate tasks, or classification models can label obstacles by their severity before these values are passed to the MCDM stage. Similarly, clustering techniques may identify groups of tasks with similar characteristics, enabling a more efficient structuring of the decision problem [112].

Thus, ML does not replace optimization or MCDM in prioritization but extends their capabilities by providing data-driven estimates of criteria or by learning hidden preference structures. This hybrid approach ensures that prioritization remains both theoretically sound and practically adaptive.

The shift toward data-driven and adaptive prioritization algorithms reflects a broader trend in robotics: the move from rigid automation toward intelligent autonomy. As systems become more capable of adjusting in real-time and learning from experience, their performance in uncertain or high-stakes environments improves significantly.

Urgency refers to the degree of time sensitivity of a task, indicating whether immediate attention is required. *Importance* is based on the degree to which a task aligns with the objectives of the overarching mission. Meanwhile, resource requirements account for the computational, sensory, or mechanical capabilities necessary to carry out the task [71]. Together, these criteria offer a structured framework to help robots determine which tasks should take precedence and how to allocate system resources efficiently.Table 3.2 presents a general categorization of robotic tasks in various domains.

3.2.2 Challenges of Decision-Making

Real-time decision making in robotics has traditionally been formulated as a multicriteria optimization problem, where tasks and actions are evaluated according to measurable objectives such as time, energy, safety,

Table 3.2: ML-Enhanced Robotic Task Categorization Across Domains [169, 30, 15]

Category	Task Description
Manufacturing and Industrial Applications	
Carry	Autonomous transport using SLAM and path optimization.
Assembly	Vision-guided precision placement via reinforcement learning.
Welding	Trajectory control with sensor feedback and CNN-based assessment.
Painting	Defect detection with visual CNNs; spray control via PID tuning.
Palletizing	Grasp planning and bin-packing via deep Q-learning.
Handling	Tactile sensing with ML classifiers for fragile object manipulation.
Packaging	Motion planning integrated with real-time object segmentation.
Control	Quality assurance via multi-sensor fusion and anomaly detection.
Inspection and Surveillance	
Inspection	Feature extraction, defect classification using supervised models.
Surveillance	Real-time scene monitoring using drones, CNN-based threat detection.
Healthcare and Emergency Response	
Surgery	Robot-assisted procedures with force-feedback and image-guided ML control.
Search/Rescue	Environment mapping and victim localization via sensor fusion and RL.
Agricultural and Environmental Tasks	
Agriculture	Yield prediction and crop monitoring via remote sensing and CNNs.
Cleaning	Obstacle-aware path planning in dynamic spaces using RL.
Education and Entertainment	
Entertainment	Human-robot interaction enhanced with emotion recognition and natural language processing.
Education	Adaptive tutoring via educational robotics and curriculum-aware ML agents.
Service and Interaction	
Customer Service	Multimodal interaction using natural language processing and intent recognition models.
Exoskeletons	Movement assistance and real-time ML control.
Autonomous Mobility	
Autonomous Driving	Sensor fusion, behavior prediction, and policy learning for navigation.
Drone Operations	Autonomous flight and visual tracking using SLAM and deep learning.

or mission value [10]. This provides a rigorous mathematical basis for prioritization, but it also reveals key challenges: criteria may be conflicting, uncertain, or only partially quantifiable. In evolving contexts, the decision space changes over time, which increases the computational burden of maintaining consistent rankings.

Another difficulty is the need to incorporate uncertainty and incomplete information. While optimization-based frameworks allow for chance constraints or probabilistic modeling, the practical estimation of probabilities and costs is non-trivial [134]. This often requires approximations, surrogate models, or domain-specific heuristics.

Machine learning can complement these methods by supplying estimated parameters or by inferring preferences when explicit models are unavailable. For example, supervised models can estimate obstacle damage levels that feed into a risk-aware criterion, or preference learning can capture operator judgments and integrate them into MCDM [84]. However, ML alone is not sufficient for decision making: without the normative structure of optimization, results risk becoming ad hoc or inconsistent.

In summary, the main challenge is to design decision frameworks that preserve the rigor of MCDM while benefiting from the adaptability of ML-based estimates. Hybrid approaches, where ML informs the inputs to formal optimization models, represent a promising direction for balancing theoretical soundness with practical flexibility.

In parallel, ML has gained traction as a means of improving real-time adaptability. By learning from data, ML models can identify relevant patterns, adapt to new conditions, and make faster, context-sensitive decisions [217]. These models excel in situations where predefined rules fall short, as they can adjust over time based on new information.

The combination of optimization and ML represents a shift towards more intelligent and resilient decision-making systems in robotics [204]. These hybrid approaches aim to balance the structure and clarity of optimization with the flexibility and learning capacity of data-driven techniques, equipping robots to handle the complexity and variability of real-world operations.

3.2.3 Algorithms to Prioritize Tasks

Algorithms for task prioritization in robotics are typically grounded in MCDM and optimization frameworks. These methods allow competing tasks to be ranked or ordered according to formalized criteria such as urgency, energy consumption, and mission relevance [28, 134]. Classical approaches include weighted sum models, outranking methods, and linear programming, all of which ensure transparent and mathematically consistent prioritization.

Machine learning contributes in a supportive manner, mainly by providing estimates of input parameters or by inferring preference structures from empirical data. For instance, ML can approximate the expected energy cost of alternative tasks, classify tasks into risk levels, or learn operator preferences from demonstrations [84]. These outputs are then fed into the MCDM stage, preserving the normative decision-analytic structure while enriching it with data-driven insights.

The combined use of MCDM and ML thus yields prioritization strategies that are both theoretically robust and practically adaptive. While MCDM ensures consistency with decision science, ML increases flexibility by refining criteria in contexts where explicit models are incomplete or uncertain.

A key component of these adaptive systems is the use of continuous feedback. Feedback loops enable the robot to assess how well tasks are executed and refine its decision-making process accordingly [42]. As the robot gains more experience, it can recognize which strategies were effective and which were not, informing future actions. This iterative learning cycle allows the system to improve task management over time, particularly in complex environments or environments in constant flux [128].

A simplified illustration of MCDM-based task prioritization can be given by considering a robot with three tasks T_1, T_2, and T_3, which need to be scheduled according to three criteria: *urgency* (U), *energy cost* (E), and *mission criticality* (M). Each criterion is normalized between 0 (lowest) and 1 (highest). A weighted sum model can compute a priority score P_i for each task:

$$P_i = w_U U_i + w_E(1 - E_i) + w_M M_i, \quad i \in \{1, 2, 3\} \tag{3.4}$$

where w_U, w_E, and w_M are operator-defined weights that sum to 1. A higher score P_i indicates a higher priority.

Table 3.3: Characteristic Example of Task Scores and Priorities

Task	Urgency U	Energy Cost E	Mission Criticality M	Priority Score P
T_1	0.9	0.6	0.8	0.86
T_2	0.5	0.3	0.9	0.77
T_3	0.7	0.8	0.4	0.64

This table illustrates how tasks are ranked based on multiple criteria. In practice, continuous updates from sensor feedback or ML predictions can adjust the input values (U_i, E_i, M_i) and the weights (w_U, w_E, w_M), enabling dynamic and adaptive prioritization in real-world operations [160, 42].

3.2.4 Human-Robot Interaction

Human-robot interaction (HRI) plays a central role when task prioritization requires input from human operators. In such settings, the decision-making problem can be represented as a bi-criteria structure:

$$C = \alpha \cdot J_{\text{robot}} + (1 - \alpha) \cdot J_{\text{human}}, \tag{3.5}$$

where J_{robot} and J_{human} are normalized to a common scale $[0, 1]$ prior to combination. Here, J_{robot} denotes the autonomous evaluation (e.g., energy, time, damage), J_{human} reflects operator preferences or safety constraints, and $\alpha \in [0, 1]$ regulates the balance between autonomy and supervision [85, 193].

One challenge is designing interfaces that allow humans to express preferences in real-time without overloading the communication channel. Studies in disaster response and medical robotics confirm that operator feedback can be modeled as additional constraints or as preference scores that modify the ranking produced by multicriteria decision-making (MCDM) [67].

Formally, let $\mathcal{A}$ denote the action set and $\succ_h$ the human-provided preference relation. The robot executes

$$a^\star = \arg\min_{a \in \mathcal{A}} C(a) \quad \text{s.t. } a \succ_h \text{ is satisfied,} \tag{3.6}$$

ensuring that critical operator instructions always dominate autonomous evaluation.

This integration highlights that HRI is not merely an interface problem, but a structured decision fusion problem: the robot must reconcile autonomous optimization with external human input. Such hybrid frameworks increase transparency and improve trustworthiness while keeping the mathematical consistency of the underlying prioritization model.

3.2.5 Conflict Scenarios

In realistic missions, robots often face conflicts where two or more high-priority tasks compete for limited resources. A general formalization is the following: let $\mathcal{T} = \{t_1, \ldots, t_m\}$ be the task set, each task t_i associated with utility u_i and resource demand r_i. For a resource budget R, a feasible set of tasks $\mathcal{F} \subseteq \mathcal{T}$ must satisfy

$$\sum_{t_i \in \mathcal{F}} r_i \leq R. \tag{3.7}$$

The decision problem is to select

$$\mathcal{F}^\star = \arg\max_{\mathcal{F} \subseteq \mathcal{T}} \sum_{t_i \in \mathcal{F}} u_i \tag{3.8}$$

subject to operational safety constraints. This formulation is analogous to a multicriteria knapsack problem with dynamic updates in real-time [150, 137].

Conflicts typically arise in three domains:

- **Safety vs. Mission Value:** For example, avoiding an obstacle may force the robot to abandon an ongoing measurement.
- **Precision vs. Resource Availability:** Performing a high-accuracy scan consumes excessive energy.
- **Urgency vs. Continuity:** Responding to a new communication window interrupts the current navigation plan.

Resolution strategies include:

1. *Optimization-based arbitration:* rank tasks using utility or lexicographic rules.
2. *Anytime scheduling:* apply algorithms that produce progressively better solutions within time limits.
3. *Learning-based adaptation:* use reinforcement learning (RL) or preference learning to approximate human-like trade-offs under uncertainty [250].

These scenarios highlight the necessity of task arbitration mechanisms that can adapt in real-time, incorporate feedback, and reason about both short-term safety and long-term mission objectives. Table 3.4 summarizes these conflict scenarios and resolution approaches.

Table 3.4: Formalization of Conflict Scenarios and Resolution Approaches [137, 150]

Conflict Scenario	**Formal Model**	**Resolution Approach**
Urgent obstacle avoidance vs. ongoing scientific task	$\max\{U_{\text{sci}}, U_{\text{safe}}\}$ with constraint $P(\text{collision}) \leq \varepsilon$	Chance-constrained optimization or RL for adaptive trade-offs
High-precision measurement vs. limited energy	$\max U_{\text{meas}}$ s.t. $E \leq \bar{E}$	Utility-based optimization; lexicographic ordering (safety $\succ$ accuracy)
Communication window vs. navigation routine	$\max(\lambda_1 U_{\text{comm}} + \lambda_2 U_{\text{nav}})$	Hybrid: hard rule for communication + RL for navigation adjustment
Sensor calibration vs. real-time risk detection	$\max(U_{\text{risk}} - \delta \cdot U_{\text{delay}})$	Anytime scheduling; heuristic arbitration

U_{sci} = utility of scientific task; U_{safe} = utility of safety action; $P(collision)$ = probability of collision; ε = acceptable risk threshold; U_{meas} = utility of measurement task; E = energy consumption; $\bar{E}$ = available energy budget; U_{comm} = communication utility; U_{nav} = navigation utility; λ_1, λ_2 = weighting factors; U_{risk} = utility of risk detection; U_{delay} = penalty for calibration delay; δ = trade-off parameter.

3.3 ML in Decision-Making

Decision making in autonomous robotics is typically modeled as a sequential optimization problem under uncertainty. A formal representation uses a MDP, defined by the tuple

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle, \tag{3.9}$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s'|s,a)$ are the transition probabilities, $R(s,a)$ is the reward function, and $\gamma \in [0,1]$ is the discount factor. The objective is to find a policy $\pi(a|s)$ that maximizes the expected return

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] \tag{3.10}$$

In this framework, RL provides algorithms to approximate optimal policies without prior knowledge of P or R [101]. Classical examples include Q-learning [170],

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \Big(R_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \Big) \tag{3.11}$$

and policy gradient methods [27], where the policy is parameterized by θ and updated via

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \big[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a) \big] \tag{3.12}$$

While optimization and MCDM provide the principal structure for robotic decision making, RL can be employed as a complementary tool in dynamic or uncertain environments. Specifically, RL is effective in:

- **Adaptive policy learning:** when explicit models of P or R are not available [171].
- **Preference approximation:** learning operator or mission preferences from interaction data [217].
- **Risk-sensitive navigation:** adjusting route selection based on observed hazards or costs [115].

Thus, ML—and RL in particular—does not replace formal decision analysis but extends it by learning approximate models or policies that improve adaptability. This hybrid perspective ensures that decision-making frameworks remain mathematically sound while being capable of handling uncertainty in real-world robotic applications.

3.3.1 Multi-Agent Reinforcement Learning (MARL)

When multiple robots operate simultaneously, decision making can be modeled as a stochastic game or Markov game, defined by the tuple [250]:

$$\mathcal{G} = \langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, \mathcal{P}, \{R_i\}_{i\in\mathcal{N}}, \gamma \rangle \tag{3.13}$$

where $\mathcal{N}$ is the set of agents, $\mathcal{S}$ the global state space, $\mathcal{A}_i$ the action space of agent i, $\mathcal{P}$ the transition function, R_i the reward function of agent i, and γ the discount factor. Each agent i seeks a policy $\pi_i(a_i|o_i)$ that maximizes its expected cumulative reward:

$$\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, a_t^1, \ldots, a_t^N) \right]. \tag{3.14}$$

Different information structures yield:

- *Centralized policies:* agents have access to joint states and actions.
- *Decentralized policies:* agents rely only on local observations.
- *Centralized training with decentralized execution:* [132].

Numerical illustration: Consider two robots traversing a corridor of 4 nodes toward opposite goals. Each agent chooses $a \in \{\text{GoForward}, \text{Wait}\}$. Rewards are defined as:

$$R_1 = \begin{cases} +10 & \text{if goal reached,} \\ -20 & \text{if collision occurs,} \\ -1 & \text{otherwise.} \end{cases} \tag{3.15}$$

Agents apply Q-learning updates:

$$Q_i(s_t, a_t^i) \leftarrow Q_i(s_t, a_t^i) + \alpha\big(R_i + \gamma \max_{a'} Q_i(s_{t+1}, a') - Q_i(s_t, a_t^i)\big). \tag{3.16}$$

Simulation results show that agents converge to cooperative strategies (waiting to avoid collisions), thereby improving safety and efficiency.

$\mathcal{S}$ = state space; $\mathcal{A}$ = action space; $P(s'|s,a)$ = state transition probability; $R(s,a)$ = reward associated with action a in state s; γ = discount factor; $\pi(a|s)$ = policy, i.e., probability of selecting action a in state s; $Q(s,a)$ = state–action value function; α = learning rate.
For MARL: $\mathcal{N}$ = set of agents; $\mathcal{A}_i$ = action space of agent i; o_i = local observation of agent i; R_i = reward function of agent i; $\pi_i(a_i|o_i)$ = policy of agent i given its observation.

MARL has diverse practical application areas. For instance, in **collision avoidance** scenarios, agents learn coordinated policies to minimize interference in confined spaces such as narrow passages. In **formation control**, they maintain group structures—e.g., specific geometric patterns for drone swarms—under dynamic conditions. In **resource sharing** problems, MARL can resolve conflicts (e.g., charging-station allocation) without centralized arbitration [58].

Despite promising results, the deployment of deep MARL in practice is often limited by data availability and training costs. Future work can integrate MARL into established frameworks for scalable, context-aware multi-robot coordination, enabling systems to learn complex collaborative strategies directly from interactions with the environment.

3.3.2 Benefits

One of the primary advantages of integrating ML into robotics is the significant boost in autonomy. Instead of relying on constant directives from human operators on Earth, robots equipped with learning algorithms can evaluate their surroundings and make decisions independently [188]. This independence is especially critical during deep-space missions, where communication delays make real-time control infeasible. Algorithms such as RL and deep neural networks enable complex functions—pathfinding, obstacle avoidance, and data collection—ultimately improving operational efficiency and reducing mission overhead.

Another notable benefit lies in the robot's ability to adapt to the unpredictable nature of extraterrestrial environments. Some planetary surfaces are often dynamic and present hazards that are difficult to anticipate, such as unstable terrain, dust storms, or extreme temperatures. Through real-time sensor analysis and continuous learning, ML models help robots detect changes in environmental conditions and adjust their behavior accordingly. For example, deep learning-based terrain classifiers can help identify traversable paths, while RL can fine-tune navigation strategies over time [87]. This adaptability is vital for maintaining the functionality and safety of the robot throughout extended missions.

ML also plays a crucial role in reducing the reliance on Earth-based operators. In many robotic missions, the round-trip communication time between the spacecraft and Earth can range from several minutes to nearly an hour. Such delays make it impractical to rely on humans for every minor decision [32]. By allowing robots to analyze their environment, assess risks, and act accordingly, ML significantly decreases latency in decision making. This capability allows the system to respond quickly to new challenges, thereby increasing mission resilience and improving the chances of success, especially in time-sensitive or high-risk scenarios.

3.4 Summary

This chapter examines the role of machine learning in autonomous systems, focusing on decision-making and task prioritization. Three main points emerged.

First, formal optimization and MCDM provide the core structure for autonomy, while ML contributes in a supporting role. In particular, RL enables adaptation in uncertain environments, and preference learning helps incorporate operator feedback into prioritization models. These methods do not replace formal decision analysis but rather extend it with data-driven estimates and learned preferences.

Second, the chapter highlighted the benefits of integrating ML into robotics: reduced reliance on human operators, greater adaptability to dynamic environments, improved use of limited resources, and increased resilience in safety-critical contexts. Each of these benefits is relevant to missions where autonomy is essential and communication is limited.

Third, potential challenges were identified, including the need for real-time operation, interpretability, and robustness under uncertainty. Solutions such as edge computing, explainable AI, and federated learning illustrate how ongoing research seeks to address these issues while preserving the reliability of decision frameworks.

Overall, ML-based methods enrich classical models of decision-making and prioritization by adding flexibility and responsiveness.

The next chapter turns to path planning, where optimization algorithms are implemented and compared to evaluate their relative strengths and limitations in realistic robotic scenarios.

Chapter 4

Evaluation of Path Planning Algorithms

So far, we have identified the literature gap, defined the problem, and reviewed the existing research in the '*Literature Review*' and '*ML Based Advances in Autonomous Systems*' chapters. This chapter presents a comparative evaluation of selected path planning algorithms to identify the most suitable one for our robotic navigation problem. Therefore, our aim is to determine the algorithm that will be employed in the following work.

Our goal is to understand each algorithm in terms of its applicability to different environments. We first examine their theoretical foundations and operational logic. Then, we assess their strengths and weaknesses under identical conditions. Given the wide variety of available path planning methods, we have limited our analysis to the most cited and widely applied algorithms in the literature. The following sections provide an overview of these algorithms, describe the selection methodology, and present a detailed evaluation of their performance. Let us start with a quick overview of path planning optimizations:

4.1 Algorithms Overview

Navigation by mobile robots is a fundamental component in autonomous systems, directly affecting task completion and safety [195]. It encompasses the strategic mapping and supervision of the robot's movement to achieve a specific objective within its surroundings. The primary goal of navigation is to allow the robot to travel securely, effectively, and independently, avoiding obstacles and reaching its target. Mobile robots are applied in domains such as autonomous vehicles, warehousing, agriculture, and search-and-rescue, with recent surveys documenting this trend [153, 50]

Given the large variety of path-planning algorithms reported in recent reviews, a controlled comparative evaluation is useful [155, 153]. The effectiveness of these algorithms can differ significantly depending on the complexity of the environment, the specific task, and various situational factors. Hence, it is crucial to evaluate algorithms in various scenarios to measure their flexibility and efficiency. This evaluation helps researchers, engineers, and professionals make well-informed decisions when selecting algorithms in practical settings.

Navigation by mobile robots has gained prominence in domains such as manufacturing and disaster response, driven by improvements in perception and planning algorithms [48]. Over the years, a variety

of techniques have been devised to facilitate robot navigation, ranging from conventional methods to more modern approaches such as ML [70].

Traditional methods for guiding mobile robots include controllers based on fuzzy logic and potential field techniques [194]. Despite their popularity for being straightforward, these techniques often face constraints. Issues such as dealing with complex situations and being vulnerable to sensor inaccuracies and uncertainties have become common challenges [199].

Learning-based approaches have been shown to handle complex and changing situations in selected settings [57]; however, they typically require substantial data and computation. These state-of-the-art methods show great potential in handling complex and changing situations. *However, they typically require significant amounts of training data and computational power.*

Algorithm assessment commonly uses criteria such as success rate, travel time, path length, and collisions [106]; we adopt a subset tailored to our setup. The performance of these algorithms is measured using different criteria, such as the success rate, duration of navigation, path distance, and collision occurrences. These assessment standards are commonly used to analyze the strengths and weaknesses of various algorithms [106].

Algorithm performance is environment-dependent; e.g., methods that optimize path length may incur higher compute time [65]. For example, although algorithm A may provide better results than algorithm B, the drawback is that algorithm A requires more processing time. *In this case, it is essential to carefully consider the main priority of the system to make the best algorithmic decision.* If time becomes a crucial aspect leading to fast results, the preferable option would be algorithm B.

Next, we will determine the efficient algorithms by examining their benefits and limitations. To facilitate comparison, the algorithms were tested under identical conditions, and their performance was assessed based on their proximity to the optimal path and computational efficiency. The subsequent section offers the Problem Formulation. The section also discusses the algorithm selection process, the methodology employed, and the tests executed, leading to a thorough evaluation of the results.

Common Notation

Input: Workspace $\mathcal{W} \subset \mathbb{R}^2$, free space $\mathcal{C}_{\text{free}}$, obstacles $\mathcal{O}$, start q_{start}, goal q_{goal}.
Output: Path $P = \{q_0, \ldots, q_n\}$ (collision-free if feasible).

4.2 Formulation of Path Planning

The path planning problem aims to determine an optimal route for a robot to travel from a start position to a goal position while avoiding obstacles. The environment may contain static and dynamic obstacles, and the solution must ensure collision-free navigation while minimizing a chosen cost metric, such as distance, travel time, or energy consumption.

Environment and Search Space Definition: Let the environment be a bounded two-dimensional space:

$$\mathcal{E} \subset \mathbb{R}^2 \tag{4.1}$$

containing a finite set of obstacles:

$$\mathcal{O} = \{o_1, o_2, \ldots, o_k\} \tag{4.2}$$

where each o_i is a polygonal region representing an obstacle.

The navigable space is defined as:

$$\mathcal{F} = \mathcal{E} \setminus \bigcup_{i=1}^{k} o_i \tag{4.3}$$

A robot R starts at $s \in \mathcal{F}$ and aims to reach a goal $g \in \mathcal{F}$.

$\mathcal{E}$: environment, $\mathcal{O}$: set of obstacles, $\mathcal{F}$: free space, s, g: start and goal positions.

These definitions in Eqs.4.1–4.3 set the continuous workspace used throughout the chapter.

Obstacle Avoidance Constraint: The planned path must remain entirely within the free space:

$$\mathcal{P}(t) \in \mathcal{F}, \quad \forall t \in [0, 1] \tag{4.4}$$

where $\mathcal{P} : [0, 1] \to \mathcal{E}$ is a continuous path from s to g with:

$$\mathcal{P}(0) = s, \quad \mathcal{P}(1) = g \tag{4.5}$$

$\mathcal{P}(t)$: continuous path from s to g, $t \in [0, 1]$: path parameter, $s, g \in \mathcal{F}$: start/goal.

The feasibility condition in Eq. 4.4 together with the boundary condition in Eq. 4.5 enforces collision-free motion from s to g. In a discrete graph-based setting, the path is represented as:

$$P = \{v_1, v_2, \ldots, v_n\}, \quad v_1 = s,\ v_n = g,\ v_i \in \mathcal{F} \tag{4.6}$$

and each edge must satisfy:

$$(v_i, v_{i+1}) \cap \mathcal{O} = \emptyset \tag{4.7}$$

meaning no path segment intersects with any obstacle.

$\mathcal{P}(t)$: continuous path, P: discrete path, v_i: waypoints, (v_i, v_{i+1}): path segment.

Eqs. 4.6–4.7 are the graph-based counterparts of the continuous feasibility in Eq. 4.4.

Optimization Formulation: The path planning problem can be expressed as:

$$\min_{\mathcal{P}} J(\mathcal{P}) \tag{4.8}$$

subject to:

$$\mathcal{P}(t) \in \mathcal{F}, \quad \forall t \in [0, 1] \tag{4.9}$$

where $J(\mathcal{P})$ is the cost function.

In continuous space:

$$J(\mathcal{P}) = \int_0^1 \left\| \frac{d\mathcal{P}(t)}{dt} \right\| dt \tag{4.10}$$

which represents the total path length.

In discrete space:

$$J(P) = \sum_{i=1}^{n-1} \text{Cost}(v_i, v_{i+1}) \tag{4.11}$$

where $J(P) = \sum_{i=1}^{n-1} \text{Cost}(v_i, v_{i+1})$ can reflect distance, time, or other mission-specific metrics.

$J(\cdot)$: objective function, $\|\frac{d\mathcal{P}(t)}{dt}\|$: instantaneous speed, Cost(v_i, v_{i+1}): transition cost between waypoints.

The continuous path-length objective in Eq. 4.10 and its discrete analog in Eq. 4.11 are the single-objective baselines used for comparison (see also Eq. 4.8).

In what follows, we consider the following planners: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Potential Field (APF), Probabilistic Roadmap (PRM), Rapidly Exploring Random Tree (RRT), fuzzy logic-based algorithms (FLs), and A* (A-star). After first mention, we use only the abbreviations.

4.3 Selection of Methods

No single planner dominates across objectives and environments; selection should align with task priorities [194, 138]. Various factors should be considered depending on the specific robotic task at hand, such as minimizing the path length, improving computational efficiency, or ensuring the reliability of the algorithm in unpredictable circumstances. An algorithm that excels in one context may not perform as effectively in another. Hence, it is crucial to align the algorithm with the objectives of the task [194, 138]. *This highlights the importance of fully understanding the strengths and weaknesses of each algorithm, emphasizing that algorithm evaluation is crucial to making well informed decisions.*

The traditional process of path planning for mobile robots typically consists of two primary stages: generating a representation of the environment and designing a path that aims to either minimize a cost function or meet specific requirements. This complex procedure involves the use of a variety of algorithms. To gain insight into this evolving field, this research focuses on seven well-known algorithms that have been widely adopted and acknowledged based on the approach outlined below:

To ground our selection, we considered algorithms that are most frequently discussed in recent surveys and reviews [153, 155]. We have used cross queries to cover all works such *as 'Path Planning', 'Robot Navigation', 'Motion Planning', 'Route Optimization'* , and so on. *Although the numbers and order have changed over time*, the list of algorithms mentioned below has been cited 9090 times out of 51104 works as of the latest check. The path planning algorithms frequently cited are ranked from most to least in Table 4.1:

We have thoroughly assessed and analyzed the selection of path planning algorithms, allowing us to make informed choices. In scenarios where finding the shortest route is crucial, we opt for the A-star algorithm due to its effectiveness and ability to ensure optimal results when coupled with a suitable heuristic [198].

For optimization tasks involving multiple objectives, GAs are effective in finding near-optimal solutions. GAs represent each potential solution as a chromosome and iteratively improve the population through selec-

No	Name	Number
1.	Genetic Algorithms (GAs)	2,615
2.	Particle Swarm Optimization (PSO)	1792
3.	Artificial Potential Field (APF)	1712
4.	Rapidly Exploring Random Tree (RRT)	1431
5.	A* Algorithm (A-star)	987
6.	Probabilistic Roadmap (PRM)	285
7.	Fuzzy Logic-based Algorithms (FLs)	268
	Total	9090

Table 4.1: Number of path planning algorithms cited in SCOPUS (June 2025 snapshot)

tion, crossover, and mutation. This makes it well-suited for complex problems where classical methods may struggle [93]. In uncertain and imprecise settings, we decided to utilize FLs for their decision-making capabilities grounded in linguistic rules and membership functions [186]. When it comes to real-time navigation in static environments, our preference was the APF algorithm because of its simplicity and computational efficiency [122]. PRM is typically favored for multi-query planning with offline roadmap construction; suitability in dynamic environments depends on roadmap updates and local replanning [158]. Furthermore, RRT's exploration bias enables fast feasibility in many scenarios, though path suboptimality is common [39]. The PSO algorithm was chosen for its strong optimization capability in complex, non-linear environments, offering efficient convergence to near-optimal paths even in the presence of multiple obstacles and constraints [243]. **Accordingly, this chapter reports a controlled evaluation of these planners across five static environments and discusses their applicability.**

4.4 Evaluation of Navigation Methods

In our assessment, we examined each algorithm based on established criteria to better understand their respective advantages and disadvantages in the mobile robot guide. Each navigation algorithm has unique characteristics and strengths. Although we considered various criteria in our analysis of seven algorithms, two primary factors emerge as universally relevant metrics for evaluation and comparison: *the length of the path* and *the processing time*. The length of the path is crucial because it shows the efficiency of the path, while processing time is significant for making immediate decisions during navigation. By focusing our evaluation criteria on these fundamental aspects, we can systematically compare algorithms on an equitable basis, obtaining valuable perspectives on their effectiveness and performance in diverse scenarios. This standardized methodology will facilitate well-informed decision making and help in the selection of the most appropriate algorithm for specific robotic tasks and optimization challenges.

The effectiveness of a methodology is highly dependent on the efficiency of its algorithm. The algorithm plays a key role in determining the value of the solution, the time needed for its execution, the parameters involved, and its suitability for the given problem. Path planning is a critical concern for mobile robots, with a variety of solutions available. Nevertheless, each solution is more appropriate for specific situations,

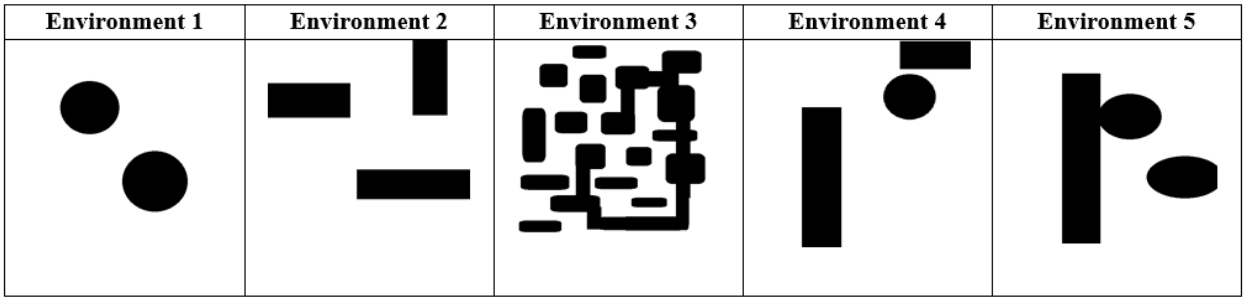


Figure 4.1: Five test environments used for algorithm evaluation.

influenced by factors such as the complexity of the environment, execution time, and path length. We will investigate the characteristics of the algorithm and identify the contexts in which it excels.

Table 4.2: Standard parameters used in test environments

Parameter	**Value**
Map Size	500×500
Obstacle Count	10
Source	(10, 10)
Goal	(490, 490)
GAs Population	50
PSO Particles	50
RRT Step Size	20
PRM Nodes	50

Our approach comprises several steps. First, we have chosen established algorithms, as elaborated in the previous section. Subsequently, five distinct static environments have been generated, as illustrated in Fig. 4.1, and the parameters for these are shown in Table 4.2. We generated five static maps with varying obstacle densities and spatial structures (Fig. 4.1); shared parameters are in Table 4.2. The focus of the comparison lies on assessing the variation in the output of algorithms within the same environment. Each algorithm is executed in these environments individually, with the starting point being the top left corner and the ending point being the bottom right corner. The results are presented in terms of *processing time* and *path length*. Finally, the results are evaluated based on the same criteria within each environment, and the findings are discussed in the conclusion section.

The results provide reproducible baselines under our settings, which can support follow-up comparisons. They can carry out their experiments using those results when selecting a path planning algorithm. This provides a consistent and repeatable structure for their studies, offering the opportunity to isolate and examine algorithmic performance without the complexities of the algorithmic details.

Our goal was to examine how well the algorithms can adjust and perform in various environments. Each environment was carefully designed to represent a different navigation challenge, allowing us to compare how well the algorithms handle a range of complexities. This methodical assessment revealed how efficient

Table 4.3: Comparison of 7 path planning algorithms across 5 environments

Algorithm	**Main Parameters**	**Values**	**Description**
A-star	Connectivity, heuristic function	8 connected grid, Euclidean heuristic	Classic deterministic method, optimal with admissible heuristic
FLs	FIS inputs/outputs, turn limit	6 inputs, 60° turn, $S = 10$	Rule based behavior adapting to obstacles and goal direction
GAs	Generations, population, waypoints	10 gens, 50 pop, 3 points, spline	Path encoded as genome, penalizes collisions, smoothing applied
PSO	Swarm size, iterations, waypoints	50, 10, 3 points, spline	Path encoded in particles, cost based fitness, smoothed output
APF	Attractive/repulsive gains, k, turn limit	3×10^5, $k = 3$, 10°, $S = 10$	Real time gradient based navigation, may suffer local minima
RRT	Step size, goal bias, threshold	20, 50% bias, 10^4 attempts, $d_{\text{th}} = 20$	Goal biased tree expansion, path extracted after tree is built
PRM	Sample size k, edge checking	$k = 50$, full connectivity, A-star	Randomized roadmap, A-star search over valid graph edges

the algorithms are under specific environmental conditions, helping us to identify the best performers for different situations. *In the end, this project gave us a comprehensive understanding of the strengths and weaknesses inherent in each approach.*

Simulation offers repeatability and safety for benchmarking and parameter sweeps [15]. One of the key benefits of employing a simulated environment is the ability to manage and reproduce various scenarios that are challenging to replicate in actual settings. This allows individuals to assess and authenticate their algorithms in a secure and regulated setting, eliminating the possibility of damaging costly equipment or causing harm to individuals. Another advantage of using such an environment is the ability to adjust various environmental elements, such as obstacles, to generate different testing scenarios [15]. Consequently, algorithms can be evaluated under a variety of circumstances, and this can be accomplished in real-world settings. These computer-based environments will be used to plan the route. The complete list of parameters for these algorithms is shown in Table 4.3. Now, let us start with the performance of the first algorithm in those environments.

4.4.1 A-star Algorithm

The A-star algorithm, commonly known as **A***, is a classic graph search method introduced in 1968 [62]. It extends Dijkstra's algorithm by combining the actual path cost with a heuristic estimate, thereby guiding the search more efficiently [198]. For each node, A-star computes the cumulative cost from the start node and a heuristic cost to the goal, expanding nodes with the lowest combined value. With an admissible and consistent heuristic, A-star is optimal and complete; it is widely used in robotics and games [62, 198, 234]. Fig. 4.2 shows the algorithm flow.

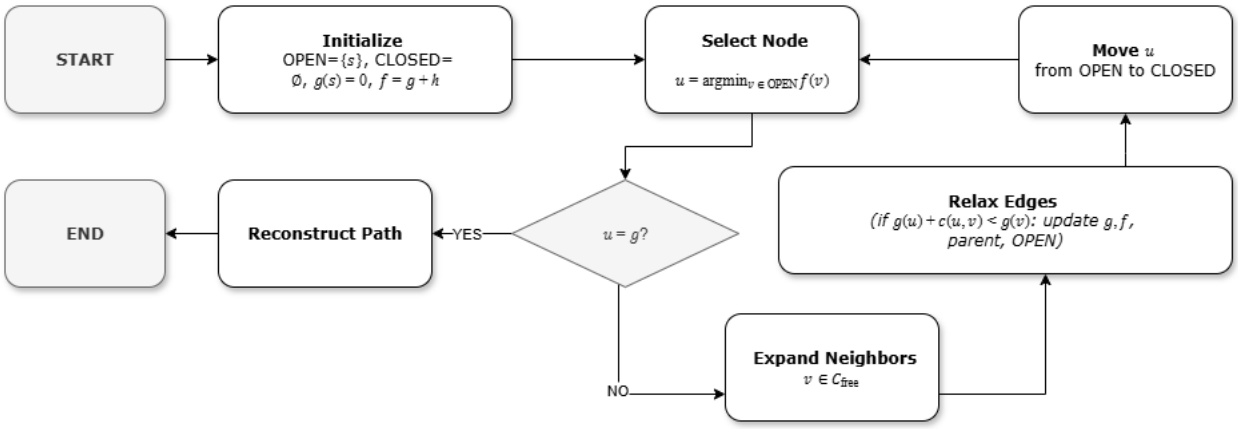


Figure 4.2: Flow diagram of the A-star.

When the heuristic is admissible and consistent, A-star guarantees optimality. Computation grows with state connectivity and poor heuristics, which can be limiting in large or dense grids [119]. Moreover, since A-star assumes a static environment, it is less suitable for dynamic scenarios where obstacles or goals change frequently, as recomputation may become computationally expensive. Although the following only presents a basic outline of the A-star algorithm, the full pseudocode is given in Algorithm C.9 in Appendix C.

Algorithm 4.1 A-star Search

Input: Graph $G = (V, E)$ derived from $\mathcal{C}_{\text{free}}$, start s, goal g

Output: Path P or FAILURE

OPEN$\leftarrow \{s\}$, CLOSED$\leftarrow \emptyset$, $g(s) \leftarrow 0$, $f(v) \leftarrow g(v) + h(v)$
while OPEN $\neq \emptyset$ **do**
  $u \leftarrow \arg\min_{v \in \text{OPEN}} f(v)$
  if $u = g$ **then return** ReconstructPath(u)
  end if
  Move u from OPEN to CLOSED
  for each v neighbor of u in $\mathcal{C}_{\text{free}}$ **do**
    if $g(u) + c(u, v) < g(v)$ **then**
      $g(v) \leftarrow g(u) + c(u, v)$, $f(v) \leftarrow g(v) + h(v)$, parent$(v) \leftarrow u$
      Insert/update v in OPEN
    end if
  end for
end while
return FAILURE

Formulation: In the discrete graph representation of the environment $G = (V, E)$, the set V contains the vertices, and E contains the edges that connect them. The objective is to determine the optimal route from the start node $s \in V$ to the goal node $g \in V$ while minimizing the total cost of travel.

The A-star algorithm is applied for this purpose due to its efficiency in heuristic-guided search. For each vertex $v \in V$, the evaluation function is defined as:

$$f(v) = g(v) + h(v) \tag{4.12}$$

where:

- $g(v)$ is the exact cumulative cost from the start node s to the current node v, computed as the sum of Cost(v_i, v_{i+1}) along the path taken so far,
- $h(v)$ is the heuristic estimate of the cost from v to the goal node g, commonly calculated using Euclidean or Manhattan distance [198].

Nodes are expanded in increasing order of $f(v)$; with an admissible and consistent $h(\cdot)$, optimality follows (cf. Eq. 4.11 for edge costs).

During the search process, the algorithm maintains an *open list* containing vertices that are candidates for exploration and a *closed list* containing vertices that have already been visited. At each iteration, the vertex with the smallest value of $f(v)$ is selected for expansion. This procedure continues until the goal node g is reached or it is determined that there is no feasible path.

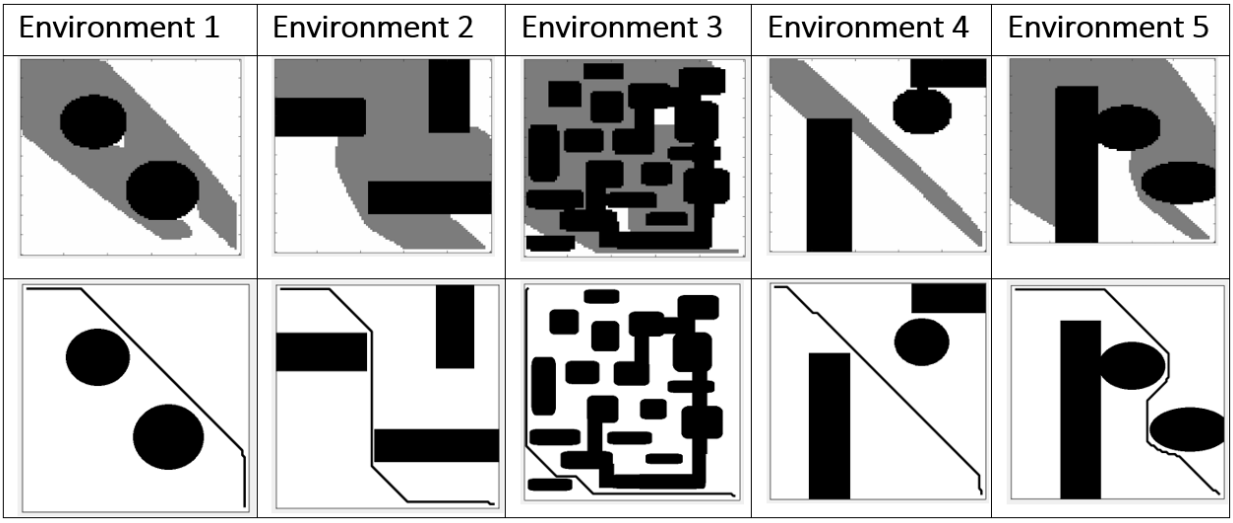


Figure 4.3: Paths generated by the A-star algorithm in all test environments.

Tests: The A-star algorithm demonstrated consistent success across all five environments, successfully finding valid paths in each scenario shown in Fig. 4.3. The algorithm was implemented on a discretized map of resolution **100×100** and utilized a user-defined connectivity matrix that allowed movements in four cardinal directions. Each node in the search carried historic, heuristic, and total cost values to guide the exploration, with the open list dynamically updated to ensure the expansion of the most promising nodes. These parameters collectively balance optimality and computational effort.

Execution times increased with environmental complexity, ranging from **5.3 s** in Environment 1 to **12.3 s** in Environment 5. This reflects the exhaustive search nature of A-star: while it guaranties an optimal or near-optimal path using an admissible heuristic, it explores a large number of nodes, which leads to longer computation times in dense or complex maps.

In terms of path length, A-star maintained efficient trajectory generation, producing paths between **702** pixels (*shortest in Environment 4*) and **899** pixels (*longest in Environment 3*). These results demonstrate the algorithm's ability to consistently identify concise and feasible routes, even under challenging conditions. The completeness and optimality of A-star make it a reliable choice for tasks that require precision in navigation. However, the increasing computational demand highlights potential limitations in real-time or resource-constrained applications. *Overall, A-star is a robust and reliable planner, in which path optimality is prioritized over execution speed, particularly suitable for structured environments where predictability and accuracy are critical.*

Snapshot

*In the map with a complex obstacle distribution, the A-star algorithm successfully computed a path with a total path length of **869 px**. The computation time was approximately **12.3 s**.*

4.4.2 Fuzzy Logic-Based Algorithms

Fuzzy logic, first introduced by Lotfi Zadeh in 1965, provides a mathematical framework for reasoning under uncertainty [238]. In mobile robot navigation, fuzzy logic employs rule-based systems with linguistic variables to support decision-making when sensor data is imprecise or incomplete.

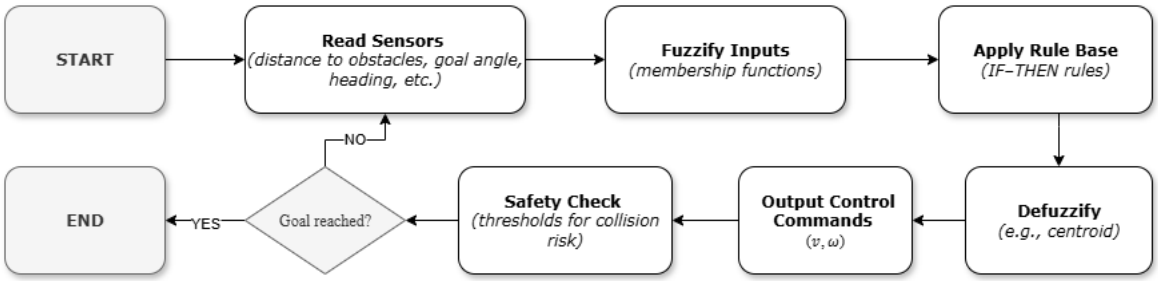


Figure 4.4: Flow diagram of the FL.

Unlike traditional binary logic, fuzzy systems allow for partial membership in sets, enabling smoother decision transitions. For instance, distance or angle readings can be expressed as "*close*," "*medium*," or "*far*" rather than as precise numerical values. Fuzzy inference rules, such *as IF the distance is close AND the angle is left THEN turn right slowly*, translate these inputs into real-time control actions [192].

The fuzzy rules used in this study are not arbitrary; they originate from a combination of expert knowledge, commonly accepted navigation heuristics in the robotics literature, and empirical observations gathered during preliminary testing. In mobile robot navigation, domain experts typically define how a robot should react when obstacles are close, when the goal is located at a specific angle, or when the robot must correct its heading. These heuristic strategies are then translated into linguistic IF–THEN statements. Several established works employ similar rule structures based on human interpretation of safe navigation behavior [199, 151].

In this thesis, the rule base was constructed by mapping intuitive decisions into fuzzy form and refining them through small-scale experiments to eliminate contradictory or ineffective rules. Therefore, the rules shown in Fig. 4.4 represent a compact, structured encoding of expert-derived and empirically validated navigation behavior.

Algorithm 4.2 Fuzzy Logic–based Navigation (the full pseudocode is given in Algorithm C.13)

Input: Sensors (distance d, goal angle θ, heading ψ), rule base

Output: Control (v, ω) till goal or FAILURE

1: **while** not at goal **do**
2: Read sensors; fuzzify inputs (d, θ, ψ)
3: Apply IF–THEN rules; infer output memberships
4: Defuzzify (e.g., centroid) $\Rightarrow (v, \omega)$
5: Send commands; safety check / emergency stop if needed
6: **end while**
7: **return** Success (END)

Formulation: The fuzzy controller employed in this thesis relies on a rule base defined over the input variables (distance to obstacle d_{obs} and goal angle θ_{goal}) to generate steering commands. Table 4.4 summarizes all IF–THEN rules used in the implementation. The linguistic terms for distance were *Near*, *Medium*, and *Far*; for goal angle *Left*, *Front*, and *Right*. The outputs correspond to steering actions: *Turn Left*, *Go Straight*, and *Turn Right*, combined with speed levels *Slow*, *Medium*, and *Fast*. These rules were used directly by the fuzzy inference engine to compute (v, ω) at each control step.

Table 4.4: Fuzzy IF–THEN rules used in the FLs navigation controller.

Distance	**Goal Angle**	**Steering Output**	**Speed Output**
Near	Left	Turn Right	Slow
Near	Front	Turn Right	Slow
Near	Right	Turn Left	Slow
Medium	Left	Turn Left	Medium
Medium	Front	Go Straight	Medium
Medium	Right	Turn Right	Medium
Far	Left	Turn Left	Fast
Far	Front	Go Straight	Fast
Far	Right	Turn Right	Fast

In the FLs navigation framework, the input variables are defined as:

- d_{obs}: distance to the nearest obstacle,
- θ_{goal}: relative angle from the robot's current position to the goal,
- ψ_{curr}: current heading of the robot.

Each input is mapped to the corresponding fuzzy sets and evaluated through a predefined rule base. The resulting fuzzy outputs are then converted into crisp control actions via a defuzzification process. Unlike the global optimization-based planners described above, the fuzzy logic controller does not compute an explicit path $\mathcal{P}(t)$ from s to g; instead, it produces real-time steering commands that incrementally guide the robot while reacting to local obstacle configurations [192].

Tests: As shown in Fig.4.5, the FLs produced a valid path only in Environment 4. It was unable to find a path in the other four environments, indicating limitations in handling complex obstacle configurations. In the successful case, the path was completed in 1.9 s with a path length of 903 px. Although the response time is low, the overall reliability in different scenarios remains limited.

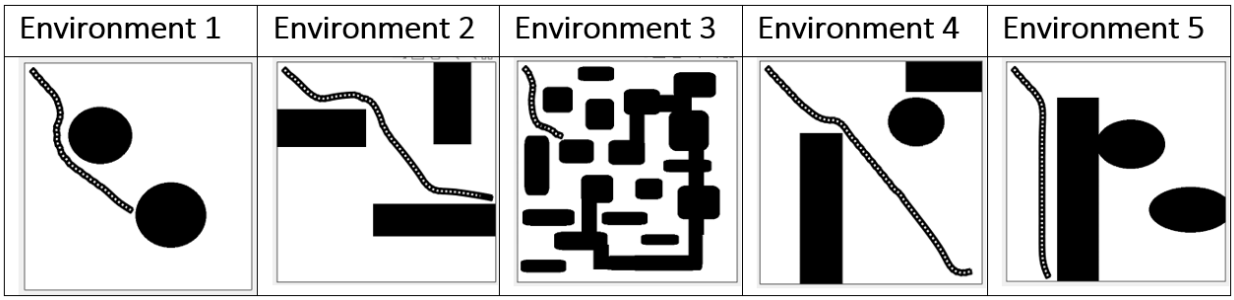


Figure 4.5: Paths generated by the FLs in all test environments.

Snapshot

*In Environment 4, the fuzzy controller generated a feasible path of **903 px** in approximately **1.9 s**. However, it did not find solutions in more complex environments.*

4.4.3 Genetic Algorithm

The GAs are population-based metaheuristics inspired by evolutionary processes, designed for high-dimensional optimization problems such as path planning. Their principle is to mimic biological evolution, where a population of candidate solutions (chromosomes) is evaluated using an objective function, and the fittest individuals reproduce through operations such as crossover and mutation. This iterative process continues until a satisfactory solution is reached [213]. GAs can explore large solution spaces and often find near-optimal paths, given sufficient generations and tuned operators [93, 144]. The general flow of GAs is shown in Fig. 4.6.

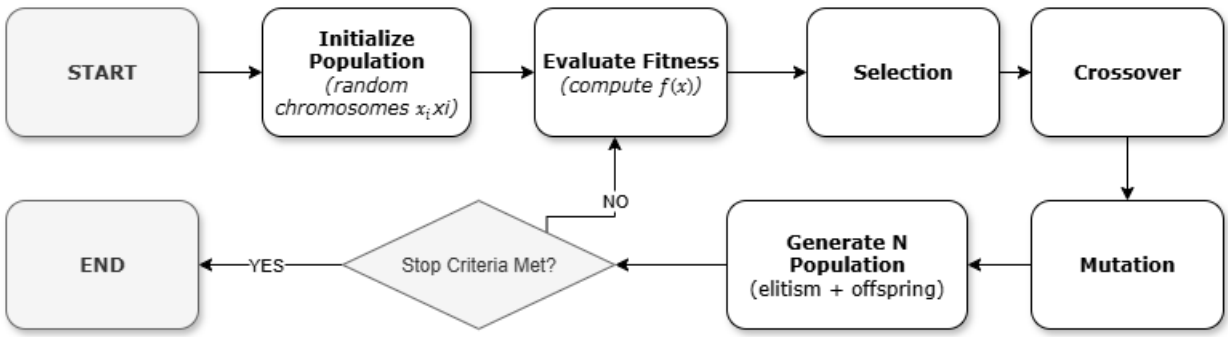


Figure 4.6: Flow diagram of the GA.

Due to their versatility, GAs are applicable to both continuous and discrete parameter spaces and are compatible with parallel processing, which accelerates convergence in large-scale problems [93]. However, while GAs provide robust global search capability, they do not guaranty the discovery of the absolute optimum [144]. Their performance strongly depends on factors such as the initial population, parameter adjustment, and the design of genetic operators. In challenging scenarios, they may require significant computational resources and time [213].

In practice, GAs start by formulating the problem, defining the objective function, and specifying the essential parameters. For example, in path planning, where a route consists of multiple waypoints, the Euclidean distance between the waypoints can be summed to form the objective function [3]. *In situations where certain routes intersect with obstacles, these segments can be incorporated as penalty terms or negative contributions to the cost function.*. Below is the high-level pseudocode of the GA; the full pseudocode of the GA implementation is presented in Algorithm C.1 in Appendix C.

Algorithm 4.3 Genetic Algorithm

Input: $\mathcal{E}, \mathcal{C}_{\text{free}}, \mathcal{O}, q_{\text{start}}, q_{\text{goal}}$

Output: Path P or FAILURE

Initialize population $\{x_i\}_{i=1}^{N}$ (random waypoints); $t \leftarrow 0$
repeat
  Evaluate $f(x) = \sum \|p_{k+1} - p_k\| + \lambda \sum \text{penalty}(p_k, \mathcal{O}) + \mu \sum \|p_{k+1} - 2p_k + p_{k-1}\|^2$
  Selection (roulette/tournament) ▷ minimize f
  Crossover: $x' = \alpha x^{(A)} + (1-\alpha)x^{(B)}$, $\alpha \sim \mathcal{U}(0,1)$
  Mutation: $x' \leftarrow x' + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$; repair if infeasible
  Form next generation with elitism; $t \leftarrow t + 1$
until $t \geq t_{\max}$ or convergence
return best x^* (if feasible) $\Rightarrow P$; else FAILURE

Formulation: In the GAs approach, the environment $\mathcal{E} \subset \mathbb{R}^2$ can be represented as a discrete grid or as a continuous coordinate space. A candidate solution (path) is encoded as a chromosome, where each gene corresponds to a coordinate point $v_i \in \mathcal{F}$ or a direction of movement.

The GAs procedure consists of the following steps:

1. **Initialization:** Generate an initial population of N random paths $x \in \mathcal{P}$, where $\mathcal{P}$ denotes the set of candidate paths.
2. **Evaluation:** Compute the fitness $f(x)$ of each path using a cost function that may include:
   - Total path length $\sum_{i=1}^{n-1} \text{Cost}(v_i, v_{i+1})$,
   - Number of turns or overall smoothness,
   - Penalty terms for proximity to obstacles,
   - Goal reachability indicator.
3. **Selection:** Choose the fittest individuals for reproduction using methods such as roulette wheel, tournament, or rank selection.
4. **Crossover:** Producing offspring by combining gene segments from two parent chromosomes.
5. **Mutation:** Introduce diversity by randomly altering genes with a small probability.
6. **Replacement:** Form the new generation by selecting the best individuals from the combined set of current and newly generated candidates.

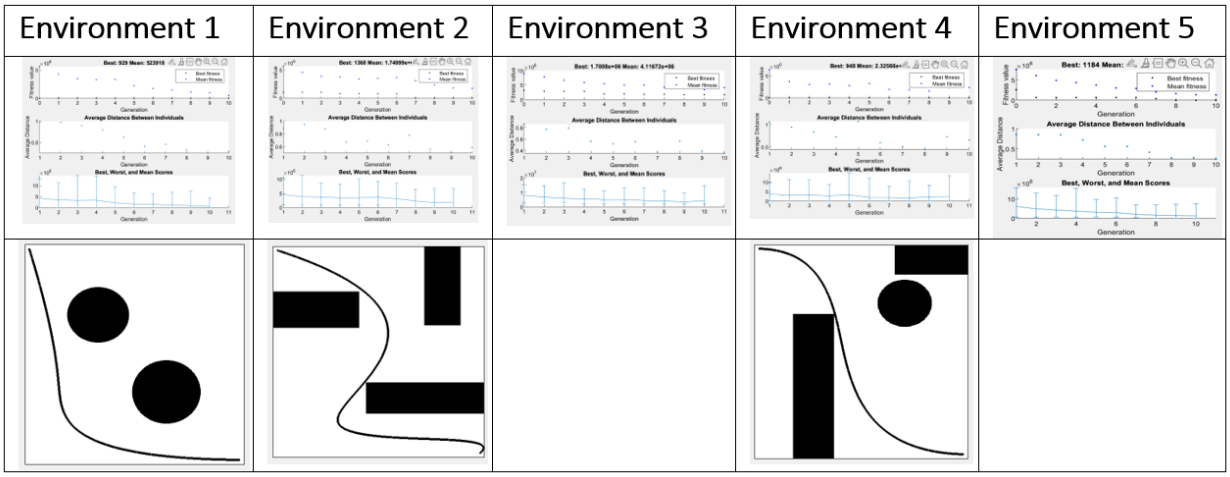


Figure 4.7: Paths generated by the GAs in all test environments.

This evolutionary cycle is repeated for a predefined number of generations or until a convergence criterion is met.

The optimization objective can be expressed as:

$$\min_{x \in \mathcal{P}} f(x) \tag{4.13}$$

where x is a candidate route in the population $\mathcal{P}$, and $f(x)$ is the fitness function that represents the total cost [213]. The chromosome-level objective in Eq. 4.13 aggregates edge costs consistent with the discrete path objective in Eq. 4.11.

In this study, the GA is configured with a population size of 50 and a maximum of 10 generations. Each chromosome encodes a path as a sequence of normalized real-valued decision variables in $[0, 1]$, which are later mapped to actual waypoint coordinates. Selection is performed using tournament selection of size 3. Crossover is implemented as an arithmetic crossover,

$$x' = \alpha x^{(A)} + (1 - \alpha) x^{(B)}, \quad \alpha \sim \mathcal{U}(0, 1), \tag{4.14}$$

applied with probability $p_c = 0.8$. Mutation is Gaussian,

$$x' \leftarrow x' + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2 I), \ \sigma = 0.05, \tag{4.15}$$

with a mutation probability $p_m = 0.1$, followed by feasibility repair to avoid collisions with obstacles. Elitism preserves the best 2 individuals in each generation. This setup ensures sufficient exploration while maintaining stable convergence behavior for environments of moderate complexity.

GAs methods are well suited for complex, high dimensional, or partially known environments because they can perform a global search and adapt to diverse problem landscapes. However, they may require higher computational resources than deterministic algorithms, particularly with large populations or detailed maps. Careful adjustment of parameters such as population size, crossover rate, and mutation rate is essential to achieve high quality results [144].

Tests: It is important to clarify the nature of the initial figures produced by the algorithm. In the first set of visualizations, the ten randomly generated decision variables of the GA are displayed using three different chart types. These variables represent the internal encoding of candidate solutions, where each chromosome is a vector of normalized values in the interval $[0, 1]$. Although they do not yet correspond to actual map coordinates, these charts illustrate how the GA initializes and perturbs individuals in the decision space. Scatter plots highlight the distribution of variables, bar charts emphasize magnitude differences, and reshaped matrices show how the same values appear when arranged in a two-dimensional structure. Their purpose is solely to reveal the structure of the search space and the representation on which crossover and mutation operators act; they are not intended to depict a real path.

Following this representation stage, the performance of GAs in the five test environments reveals a mixed and environment-dependent behavior (Fig. 4.7). The algorithm successfully found paths in Environments 1, 2, and 4, with corresponding execution times of 5.7, 2.4, and 4.0 s. However, it failed to discover any valid paths in Environments 3 and 5, which are displayed as empty. This inconsistency indicates that the GA's effectiveness is highly sensitive to the complexity and structural characteristics of the environment. In relatively open or moderately complex scenarios, such as Environments 1 and 2, the GA performs well, with Environment 2 achieving particularly fast computation. In contrast, the failure in Environment 3—a maze-like configuration—suggests that the algorithm struggles in narrow or highly constrained spaces unless its population size or iteration count is sufficiently increased.

In terms of path length, the GA produced varied outcomes: **931** px in Environment 1, a significantly longer **1710** px in Environment 2, and **913** px in Environment 4. The excessive length in Environment 2 may indicate convergence to a suboptimal solution or insufficient exploration due to parameter settings. Although GAs are recognized for their global search capabilities and adaptability, their performance depends heavily on the quality of the initial population, parameter tuning, and the number of generations. These results highlight both the potential and limitations of the GA: it *can serve as an effective optimizer under suitable conditions, but it requires careful configuration and does not guaranty success in more constrained or complex environments*.

Snapshot

*Using a population of 50 and 10 generations, the GAs found a feasible path in the 4th environment. The path length was approximately **913 px** , with a total execution time of around **4.0 s**.*

4.4.4 Artificial Potential Fields

In this method, the robot navigates by following the gradient of a potential field, where the target is modeled as an attractive potential and obstacles as repulsive potentials [122]. The gradient guides the robot from high potential regions around obstacles toward the low potential goal, enabling it to traverse collisions while progressing toward the objective. APF is computationally light and local; however, local minima and narrow-passage failures are common [122, 220]. However, it remains vulnerable to local minima, where the robot may be trapped in non-target positions [195].

APF is valued for its simplicity and effectiveness in real-time planning, making it suitable for tasks such as navigation, route mapping, and swarm coordination [220]. Its lightweight computing allows for deployment in embedded systems or scenarios that require rapid decisions. However, APF methods can struggle in narrow passages or highly cluttered spaces, and their performance degrades in dynamic environments, since continuously updating the potential field may cause additional computational overhead [60]. Fig. 4.8 illustrates the workflow of the algorithm.

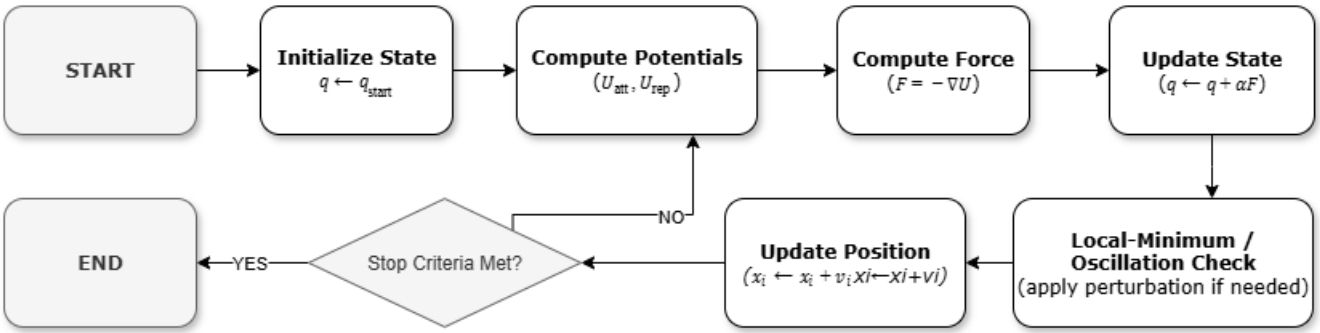


Figure 4.8: Flow diagram of the APF.

Algorithm 4.4 Artificial Potential Field (the complete pseudocode is provided in Algorithm C.4)

Input: $\mathcal{E}, \mathcal{C}_{\text{free}}, \mathcal{O}, q_{\text{start}}, q_{\text{goal}}$

Output: Path P or FAILURE

1: $q \leftarrow q_{\text{start}}$; $P \leftarrow [q]$
2: **repeat**
3: $U_{\text{att}}(q) = \frac{1}{2} k_{\text{att}} \|q - q_{\text{goal}}\|^2$
4: $U_{\text{rep}}(q) = \frac{1}{2} k_{\text{rep}} (\frac{1}{\rho(q)} - \frac{1}{\rho_0})_+^2$
5: $F(q) \leftarrow -\nabla(U_{\text{att}} + U_{\text{rep}})$; $q \leftarrow q + \alpha F(q)$
6: Append q to P; apply perturbation if trapped (local minimum)
7: **until** $\|q - q_{\text{goal}}\| \leq \varepsilon$ or max steps
8: **return** P if collision-free; else FAILURE

Formulation: In the APF method, the environment $\mathcal{E} \subset \mathbb{R}^2$ is modeled as a potential field in which the robot configuration $q \in \mathcal{F}$ is simultaneously attracted to the goal and repelled from obstacles. The total potential $U(q)$ is defined as:

$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q) \tag{4.16}$$

where:

- $U_{\text{att}}(q)$ is the *attractive potential* pulling the robot toward the goal q_{goal},
- $U_{\text{rep}}(q)$ is the *repulsive potential* that pushes the robot away from the nearest obstacle.

The attractive potential is typically expressed as:

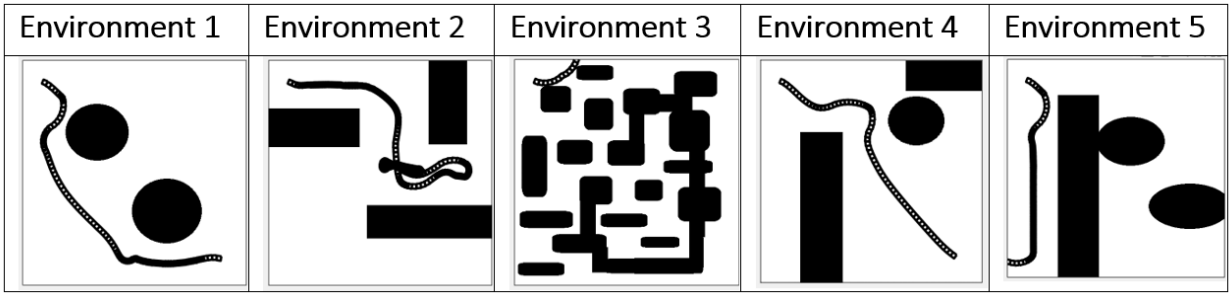


Figure 4.9: Paths generated by the APF algorithm in all test environments.

$$U_{\text{att}}(q) = \frac{1}{2} k_{\text{att}} \|q - q_{\text{goal}}\|^2 \tag{4.17}$$

where $k_{\text{att}} > 0$ is a scaling factor.

The repulsive potential is defined as:

$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2} k_{\text{rep}} \left(\frac{1}{\rho(q)} - \frac{1}{\rho_0} \right)^2 & \text{if } \rho(q) \leq \rho_0, \\ 0 & \text{if } \rho(q) > \rho_0, \end{cases} \tag{4.18}$$

where:

- $k_{\text{rep}} > 0$ is a scaling factor,
- $\rho(q)$ is the Euclidean distance from q to the closest obstacle $o_k \in \mathcal{O}$,
- $\rho_0 > 0$ is the influence range beyond which the repulsive potential becomes zero.

The force acting on the robot is the negative gradient of the total potential:

$$F(q) = -\nabla U(q) = -\nabla U_{\text{att}}(q) - \nabla U_{\text{rep}}(q) \tag{4.19}$$

The robot moves in the direction of $F(q)$ until the goal is reached. However, APF methods can suffer from *local minima* issues, where $F(q) = 0$ occurs at a point that is not the goal [220]. Navigation follows the descent direction $F(q)$ in Eq. 4.19; local minima are the main failure mode despite their efficiency.

Tests: Before discussing the results, the main parameters used in the APF implementation are summarized here, as they strongly influence the behavior of the method. The robot has a size of $[10, 10]$ px, an initial heading of $\pi/8$, and moves with a maximum speed of 10 units, with acceleration constraints of ± 10 units per step. The attractive–repulsive potential field is shaped by the exponents $k = 3$, an attractive scaling factor of 300,000, a repulsive scaling factor of 300,000, and a minimum attractive potential of 0.5. The turning angle is limited to $\pm 10^\circ$ per iteration, and the goal is considered reached when the robot enters a 30 px threshold around it. Distance measurements are obtained in five directions (front, left, right, and two diagonals), and feasibility checks ensure obstacle-free motion at each step. These constraints collectively determine the robot's reactivity and vulnerability to potential traps.

As shown in Fig. 4.9, the APF algorithm successfully found valid paths in only two of the five test environments: Environments 1 and 4. In these simpler and less cluttered layouts, the algorithm demonstrated fast execution times of **3.9** and **1.4** s, respectively. Such computation speed is typical for APF, as its gradient-based updates require minimal processing per step. The resulting paths measured **784** px and **780** px, indicating reasonably direct trajectories around obstacles.

However, in Environments 2, 3, and 5, the algorithm failed to identify feasible paths. These failures are consistent with the well-known limitations of APF methods: *their susceptibility to local minima and deadlock configurations*. In more complex scenes—such as narrow corridors, U-shaped traps, or maze-like structures—the repulsive and attractive potentials may balance out in undesired ways, causing the robot to stall or oscillate without progressing toward the goal. In Environment 3, for example, the maze-like geometry likely created strong symmetric repulsive fields, trapping the robot far from the goal despite the correct global intent. Similarly, Environment 5 contains concave regions where the attractive potential was insufficient to pull the robot out of local basins.

Overall, the results highlight that APF can be extremely efficient in simple, static environments but is unreliable in complex obstacle distributions. **APF is, therefore, best suited for open spaces with smooth potential gradients, while alternative or hybrid global planners are required for navigating cluttered or highly constrained environments.**

Snapshot

*The APF based approach computed a collision free path with a total path length of **780** px. The robot reached the goal in approximately **1.4** s at best.*

4.4.5 Probabilistic Road Map

The PRM algorithm is widely used for motion planning in complex, high-dimensional configuration spaces [158]. It constructs a roadmap of feasible configurations and valid connections, which can then be used to solve multiple planning queries. PRM is particularly advantageous in environments with moving obstacles since the roadmap is built offline and only start–goal collision checks are required. PRM accommodates complex geometries; handling nonholonomic constraints typically relies on an appropriate local planner [158, 107].

The performance of PRM strongly depends on the number and distribution of sample configurations. Sparse sampling in critical regions may prevent the discovery of feasible paths, while constructing the initial roadmap can be computationally expensive in high-dimensional spaces. Sample size and connection strategy critically affect connectivity and solution quality [152]. Refer to Fig. 4.10 to see the algorithm flow and C.11 for the complete pseudocode.

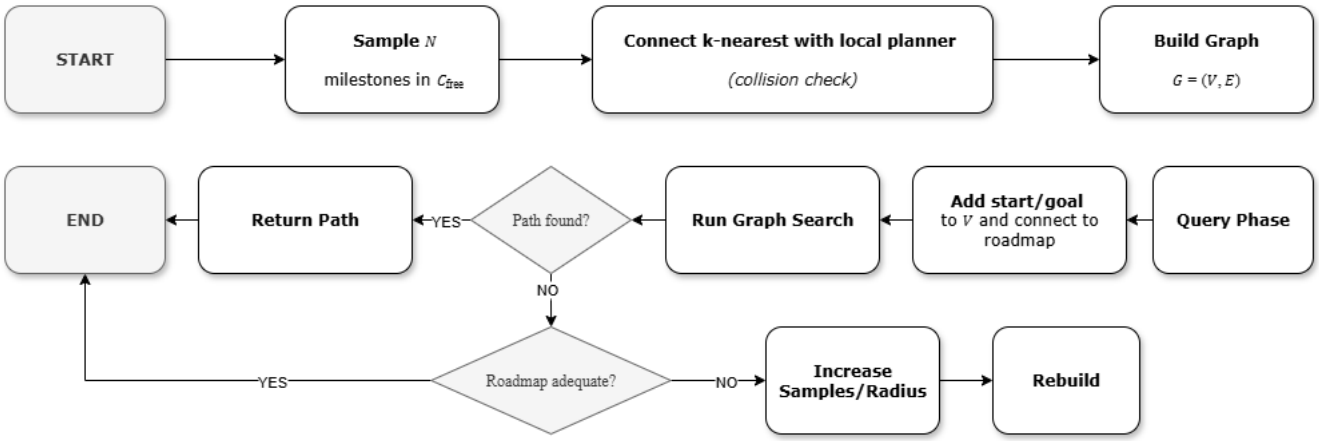


Figure 4.10: Flow diagram of the PRM.

Algorithm 4.5 Probabilistic Roadmap

Input: $\mathcal{C} = \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{obs}}$, start q_s, goal q_g

Output: Path P or FAILURE

1: **Roadmap build:** sample N milestones in $\mathcal{C}_{\text{free}}$; connect k-NN with local planner if collision-free $\Rightarrow$ $G = (V, E)$
2: **Query:** add q_s, q_g to V; connect to roadmap
3: Run graph search (Dijkstra/A*) on G
4: **if** path exists **then return** P
5: **elsereturn** FAILURE
6: **end if**

Formulation: The PRM algorithm is a two-phase sampling-based motion planning method [158]. It operates in a continuous configuration space $\mathcal{C} \subset \mathbb{R}^n$, divided into the free space $\mathcal{C}_{\text{free}}$ and the obstacle space $\mathcal{C}_{\text{obs}}$, such that:

$$\mathcal{C} = \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{obs}}, \quad \mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{obs}} = \emptyset \tag{4.20}$$

Phase 1 – Roadmap Construction:

1. Randomly sample N configurations $\{q_1, q_2, \ldots, q_N\}$ with $q_i \in \mathcal{C}_{\text{free}}$.
2. For each sample configuration q_i, identify its nearest neighbors k $\{q_{i_1}, q_{i_2}, \ldots, q_{i_k}\}$.
3. Attempt to connect q_i to each neighbor using a simple local planner, provided that the path is entirely within $\mathcal{C}_{\text{free}}$.
4. The result is an undirected graph $G = (V, E)$ where:

$$V = \{q_i\}_{i=1}^{N} \cup \{q_{\text{start}}, q_{\text{goal}}\}, \quad E = \{(q_i, q_j) \mid \text{collision free path exists between } q_i \text{ and } q_j\} \tag{4.21}$$

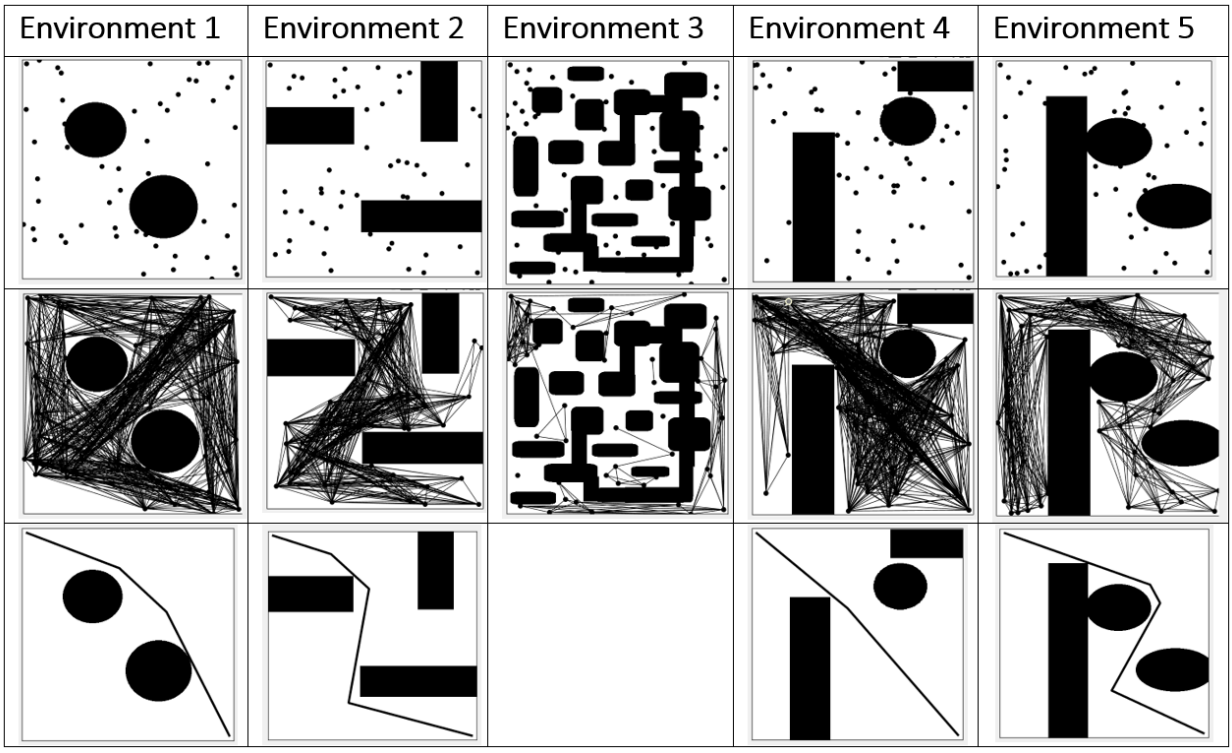


Figure 4.11: Paths generated by the PRM algorithm in all test environments.

The query phase searches G with a graph search consistent with the discrete objective in Eq. 4.11. **Phase 2 – Query/Search:**

- Add the start and goal configurations q_{start} and q_{goal} to V and connect them to the nearby nodes in $\mathcal{C}_{\text{free}}$.
- Apply a graph search algorithm (*like Dijkstra, A-star*) to find a sequence of vertices connecting q_{start} to q_{goal}.

The resulting solution is a path:

$$\mathcal{P} = \{q_{\text{start}}, q_{i_1}, q_{i_2}, \ldots, q_{\text{goal}}\} \tag{4.22}$$

such that:

$$q_j \in \mathcal{C}_{\text{free}}, \quad \forall j, \quad \text{and} \quad \min_{\mathcal{P}} \sum_{j=0}^{n-1} \|q_{j+1} - q_j\| \tag{4.23}$$

where $\|q_{j+1} - q_j\|$ represents the Euclidean distance between consecutive configurations along the path [152]. Eq. 4.23 states the feasibility and the path-length objective along the roadmap (cf. Eq. 4.11).

Tests: It is useful to summarize the parameters used in the PRM implementation, as these directly affect sampling density, roadmap connectivity, and overall performance. The roadmap size was set to $k = 50$ sampled nodes, with the source and goal appended as additional vertices to facilitate a reliable A-star search.

Nodes were sampled uniformly across the free space, and each new vertex was connected to all others through a visibility check using the `checkPath` function. The adjacency structure was represented as a cell-based list, and the global path was computed using an A* search variant in which each node stored its historic, heuristic, and total costs. The feasibility of each candidate node and each connecting edge was verified using collision checking against the binary map representation.

As depicted in Fig. 4.11, the PRM algorithm demonstrated strong and stable performance in four out of five environments, successfully finding feasible paths with relatively low computation times. The recorded execution times ranged from **3.3** to **4.2** s, indicating a favorable balance between efficiency and effectiveness. The only failure occurred in Environment 3, where the method did not produce a valid route. This behavior is consistent with the inherent limitations of PRM: if the random sampling fails to provide sufficient node density in narrow corridors, or if visibility-based connections cannot form a fully connected graph, the roadmap becomes fragmented, and the subsequent A* query cannot reach the goal. Environment 3, with its maze-like layout, likely required a higher sampling density or an increased connection radius to bridge constrained regions of free space.

In terms of path quality, PRM consistently produced some of the shortest trajectories among all the algorithms tested, with lengths ranging **from 682** to **980** px. Notably, Environment 4 yielded a path as short as **682** px while maintaining low computation time, demonstrating PRM's strength in generating efficient global plans once an adequate roadmap exists. This benefit arises from its two-stage architecture: an offline graph construction followed by a graph-based optimal search using A*. *However, the dependence on sufficiently dense and well-connected samples means that PRM can underperform in tightly constrained environments unless its sampling parameters are carefully tuned.* Overall, the results show that PRM offers an excellent trade-off between performance and path optimality, making it well suited for semi-structured or moderately complex environments where offline construction is acceptable.

Snapshot

*With $k = 50$ sampled nodes, PRM constructed a feasible roadmap and found a path of length **682 px**. The planning was completed in about **3.3 s**.*

4.4.6 Rapidly Exploring Random Trees

The RRT algorithm incrementally expands a tree by connecting random samples to the nearest existing node until the goal configuration is reached. This approach is particularly effective in high-dimensional configuration spaces and for systems subject to nonholonomic constraints [107]. The resulting structure is a branching tree, where each node corresponds to a sampled configuration, and the edges represent feasible local connections [26].

A key factor influencing performance is the branch length: longer steps reduce computational effort but may lead to suboptimal paths, while shorter steps improve path quality at the cost of higher execution time. The algorithm balances these trade-offs by biasing the sampling process towards unexplored regions, allowing for efficient space coverage [39].

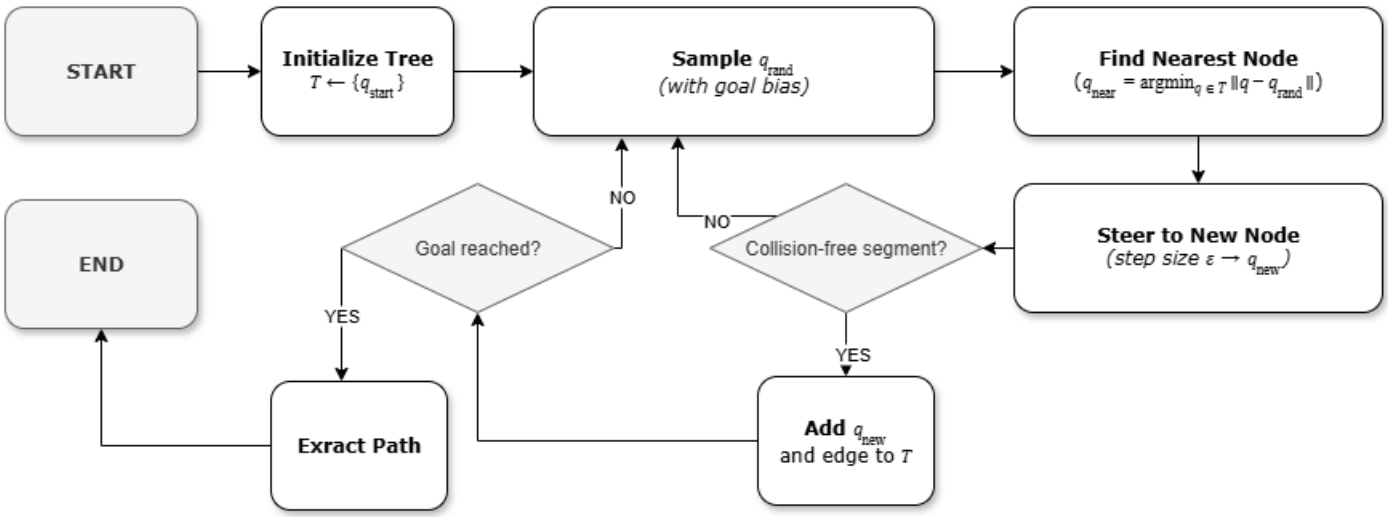


Figure 4.12: Flow diagram of the RRT.

RRT is often used for fast feasibility in complex spaces; in dynamic settings, it is combined with replanning or RRT variants [107, 39]. It has been successfully applied in various robotic tasks, such as autonomous vehicle navigation and robotic arm motion planning [195, 196]. The algorithm flow can be seen in Fig. 4.12.

Algorithm 4.6 Rapidly-Exploring Random Tree (the complete pseudocode is provided in Algorithm C.7)

Input: $\mathcal{C} = \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{obs}}$, $q_{\text{start}}, q_{\text{goal}}$

Output: Path P or FAILURE

$T \leftarrow (\{q_{\text{start}}\}, \emptyset)$
repeat
  Sample q_{rand} (with goal bias)
  $q_{\text{near}} = \arg\min_{q \in T} \|q - q_{\text{rand}}\|$
  $q_{\text{new}} = \text{steer}(q_{\text{near}}, q_{\text{rand}}, \varepsilon)$
  if segment $(q_{\text{near}}, q_{\text{new}}) \subset \mathcal{C}_{\text{free}}$ **then**
    Add q_{new} and edge to T
  end if
until $\|q_{\text{new}} - q_{\text{goal}}\| \leq \varepsilon$ or iter. limit
return ExtractPath(T, q_{goal}) if reached; else FAILURE

Formulation: The RRT is a sampling-based motion planning algorithm that incrementally builds a tree rooted in the starting configuration $q_{\text{start}} \in \mathcal{C}_{\text{free}}$, with the aim of efficiently exploring the configuration space $\mathcal{C} \subset \mathbb{R}^n$.

Algorithm Steps:

1. Initialize the tree $T = (V, E)$ with $V = \{q_{\text{start}}\}$ and $E = \emptyset$.
2. At each iteration:
   - Randomly sample a configuration $q_{\text{rand}} \in \mathcal{C}$.

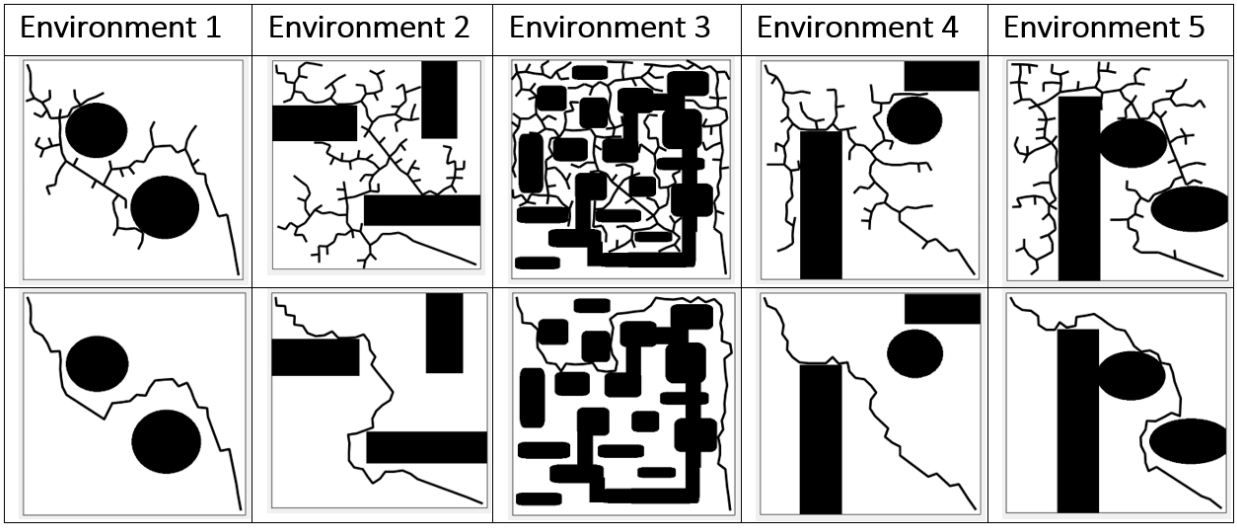


Figure 4.13: Paths generated by the RRT algorithm in all test environments.

- Find the nearest vertex in the tree:

$$q_{\text{near}} = \arg\min_{q \in V} \|q - q_{\text{rand}}\| \tag{4.24}$$

q_{rand}: random sample, q_{near}: nearest tree node, q_{new}: steered node, ϵ: step size.

- Generate a new vertex q_{new} by moving from q_{near} toward q_{rand} by a fixed step size ϵ.
- If the path segment $(q_{\text{near}}, q_{\text{new}})$ is entirely within $\mathcal{C}_{\text{free}}$, add q_{new} to V and $(q_{\text{near}}, q_{\text{new}})$ to E.

3. Repeat until q_{goal} is reached or the maximum number of iterations is exceeded.

The algorithm produces a tree $T = (V, E)$ and a feasible path:

$$\mathcal{P} = \{q_{\text{start}}, q_{i_1}, q_{i_2}, \ldots, q_{\text{goal}}\}, \tag{4.25}$$

such that

$$\forall j \in \{0, 1, \ldots, n\} : \; q_j \in \mathcal{C}_{\text{free}}, \tag{4.26}$$

and

$$\min_{\mathcal{P}} \sum_{j=0}^{n-1} \|q_{j+1} - q_j\|, \tag{4.27}$$

where $\|q_{j+1} - q_j\|$ denotes the Euclidean distance between consecutive configurations along $\mathcal{P}$ [249]. The tree induces a feasible path in Eq. 4.25; Eqs. 4.26–4.27 mirror the discrete feasibility and length objective (cf. Eqs. 4.6, 4.11).

RRT is well suited for high-dimensional or complex spaces and can handle dynamic environments. *However, it does not guaranty optimality, and the resulting paths may require post processing to improve smoothness and efficiency.*

Tests: As illustrated in Fig. 4.13, the RRT algorithm exhibited consistent success across all five environments, effectively identifying viable paths even in the more complex scenarios, such as Environment 3. This highlights the strength of RRT in rapidly exploring high-dimensional and cluttered spaces by incrementally building a tree structure toward the goal.

The algorithm was configured with a step size of **20 px**, a distance threshold of **20 px** for node uniqueness, and a maximum of **10,000 failed attempts** per iteration to ensure thorough exploration. To bias the tree toward the goal, 50% of the samples were drawn directly from the goal location. These parameters collectively balance exploration with convergence efficiency.

The recorded execution times remained relatively low in most settings; notably, **2.4** to **2.9 s** in four out of five environments, with the exception of Environment 3, where the complexity caused a significant increase to **6.4 s**. This indicates that while RRT is generally computationally efficient, densely packed obstacles and narrow corridors can temporarily hinder convergence due to increased sampling and collision-checking.

Regarding the path lengths, RRT produced trajectories ranging from **758** to **1057 px**, with the longest path occurring in Environment 3. This reflects the exploratory, non-optimized nature of RRT paths, which prioritize finding a feasible solution over optimality. Despite the longer paths, the algorithm consistently succeeded in generating valid routes, confirming its robustness in complex and high-dimensional spaces.

In general, RRT emerges as a practical option when fast feasibility and broad exploration are prioritized over path optimality, especially in applications such as robotic arms or autonomous vehicles operating in unpredictable environments.

Snapshot

*The RRT algorithm explored the space with step size 20 and reached the goal node efficiently. The best path length was **758 px**, and the total runtime was around **2.4 s**.*

4.4.7 Particle Swarm Optimization

PSO is a population-based metaheuristic inspired by the collective behavior of birds and fish, introduced by Kennedy and Eberhart in 1995 [95]. It solves optimization problems by simulating a swarm of particles that explores the search space cooperatively. A basic sequence associated with the PSO algorithm is illustrated in Fig. 4.14.

In mobile robot path planning, each particle encodes a candidate path from the start to the goal. Particles update their positions based on their own best experiences and the global best found by the swarm, gradually refining solutions toward optimal paths.

PSO is simple to implement and can handle nonlinear objectives; the convergence quality depends on the inertia and learning coefficients [95, 43, 243]. Unlike gradient-based methods, it does not require explicit models, which makes it suitable for environments with obstacles and complex cost functions.

However, PSO may converge prematurely to local minima if the diversity in the swarm is not preserved. Its performance strongly depends on parameter tuning, such as swarm size, inertia weight, and learning coefficients. In dynamic environments, additional mechanisms, such as the reinitialization of particles, are often required to maintain robustness [43].

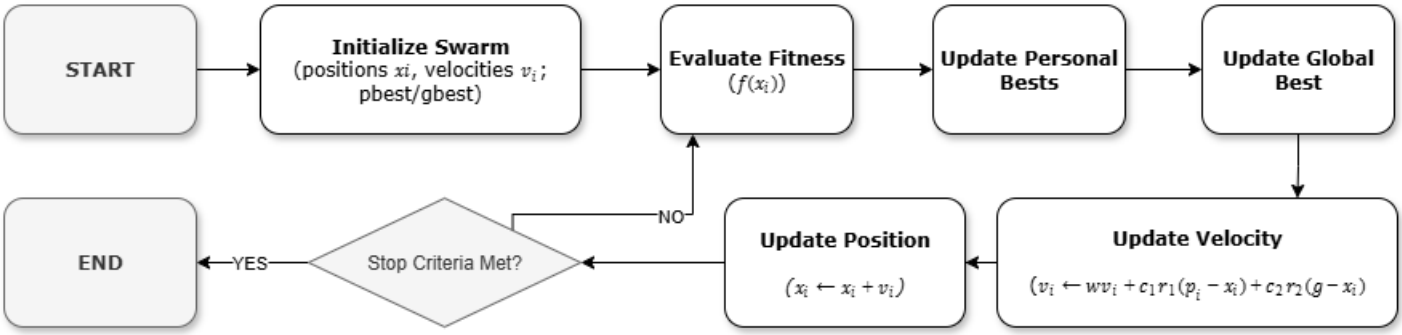


Figure 4.14: Flow diagram of the PSO algorithm.

In robotic navigation, the PSO typically represents the waypoints of the robot as particles, and the cost function incorporates the path length, smoothness, and avoidance of obstacles. Through iterative updates of positions and velocities, the algorithm converges toward feasible and efficient trajectories. PSO has been successfully applied in areas such as autonomous navigation, multirobot coordination, and real-time motion planning, demonstrating its balance between exploration and exploitation [243]. A brief outline of the PSO algorithm is provided below; for the full pseudocode, please refer to C.2.

Algorithm 4.7 Particle Swarm Optimization

Input: $\mathcal{E}, \mathcal{C}_{\text{free}}, \mathcal{O}, q_{\text{start}}, q_{\text{goal}}$

Output: Path P or FAILURE

1: Initialize swarm $\{x_i, v_i\}_{i=1}^{N}$; set $\text{pbest}_i \leftarrow x_i$, gbest
2: **repeat**
3: Evaluate $f(x_i)$; update pbest_i and gbest
4: $v_i \leftarrow wv_i + c_1r_1(\text{pbest}_i - x_i) + c_2r_2(\text{gbest} - x_i)$
5: $x_i \leftarrow x_i + v_i$; project/penalize if colliding
6: **until** stop criterion met
7: **return** gbest $\Rightarrow P$ (if feasible) else FAILURE

Formulation: In PSO, a population swarm of particles explores the configuration space $\mathcal{C} \subset \mathbb{R}^n$ to find an optimal path $\mathcal{P}$. Each particle i encodes a candidate path x_i and has an associated velocity v_i.

At each iteration t, the position and velocity of each particle are updated as:

$$v_i(t+1) = w\, v_i(t) + c_1 r_1\, (p_i - x_i(t)) + c_2 r_2\, (g - x_i(t)) \tag{4.28}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{4.29}$$

where:

- w is the inertia weight, which controls the influence of the previous velocity.
- c_1, c_2 are acceleration coefficients for cognitive and social components,

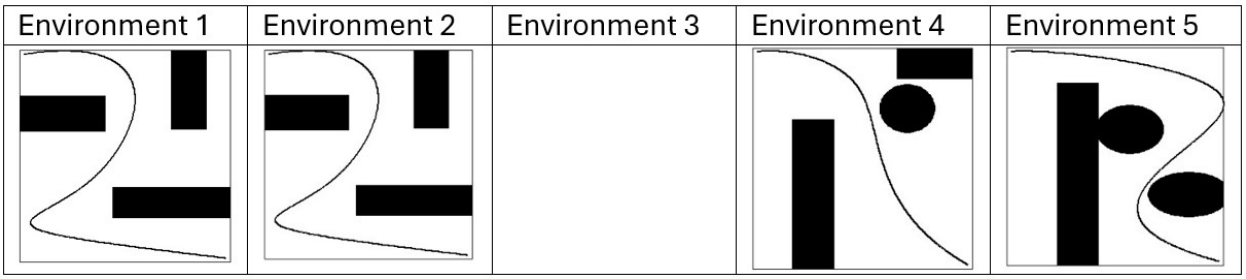


Figure 4.15: Paths generated by the PSO algorithm in all test environments.

- $r_1, r_2 \sim \mathcal{U}(0,1)$ are random numbers that introduce stochasticity,
- p_i is the best position found so far by particle i,
- g is the best global position found by the swarm.

The optimization problem is formulated as:

$$\min_{\mathcal{P}} f(\mathcal{P}) \tag{4.30}$$

where the cost function $f(\mathcal{P})$ typically combines path length, smoothness, and obstacle avoidance:

$$f(\mathcal{P}) = \sum_{j=0}^{n-1} \|x_{j+1} - x_j\| + \lambda \sum_{j=0}^{n} \text{penalty}(x_j) \tag{4.31}$$

The scalarized objective in Eqs. 4.30–4.31 combines length and collision penalties, consistent with the baseline in Eq. 4.11. Here, $\|x_{j+1} - x_j\|$ denotes the Euclidean distance between consecutive points along the path, and penalty(x_j) represents collision or proximity costs scaled by $\lambda > 0$ [243].

PSO is effective for high-dimensional, non-linear, and partially known environments, often providing near-optimal solutions with good convergence characteristics.

Tests: In Fig. 4.15, the PSO algorithm successfully found valid paths in four out of five test environments: Environments 1, 2, 4, and 5, while it failed in Environment 3, as indicated by the absence of timing and path length results. The algorithm was configured with **50 particles** in the swarm, **10 generations**, and each candidate path represented by **3 intermediate points** , excluding the source and goal. Additionally, spline-based smoothing was applied to enhance path continuity. These parameters were chosen to balance the exploration of the search space with convergence speed while maintaining path smoothness.

Execution times were relatively stable across the successful cases, with **5.5 s** in Environments 1 and 2, **5.2 s** in Environment 4, and **6.9 s** in Environment 5. This consistency indicates that PSO handled moderately complex scenarios with a predictable computational demand.

Regarding path quality, PSO generated compact and efficient trajectories, with path lengths of **733**, **836**, **717**, and **722 px** in the respective environments. These results demonstrate improved path optimality compared to earlier configurations, confirming that the algorithm can produce smooth and short routes when swarm parameters are well tuned. However, the failure in Environment 3, likely due to tighter passages or a

denser obstacle distribution, highlights the susceptibility of PSO to local optima and the importance of sufficient swarm diversity and iteration depth. *Overall, PSO presents a balanced trade-off between computation time and solution quality, making it suitable for structured to moderately complex environments where path smoothness and efficiency are prioritized.*

Snapshot

*PSO generated optimized paths using 50 particles over 10 generations. It reached the goal with a path length of **717 px** in approximately **5.8 s**.*

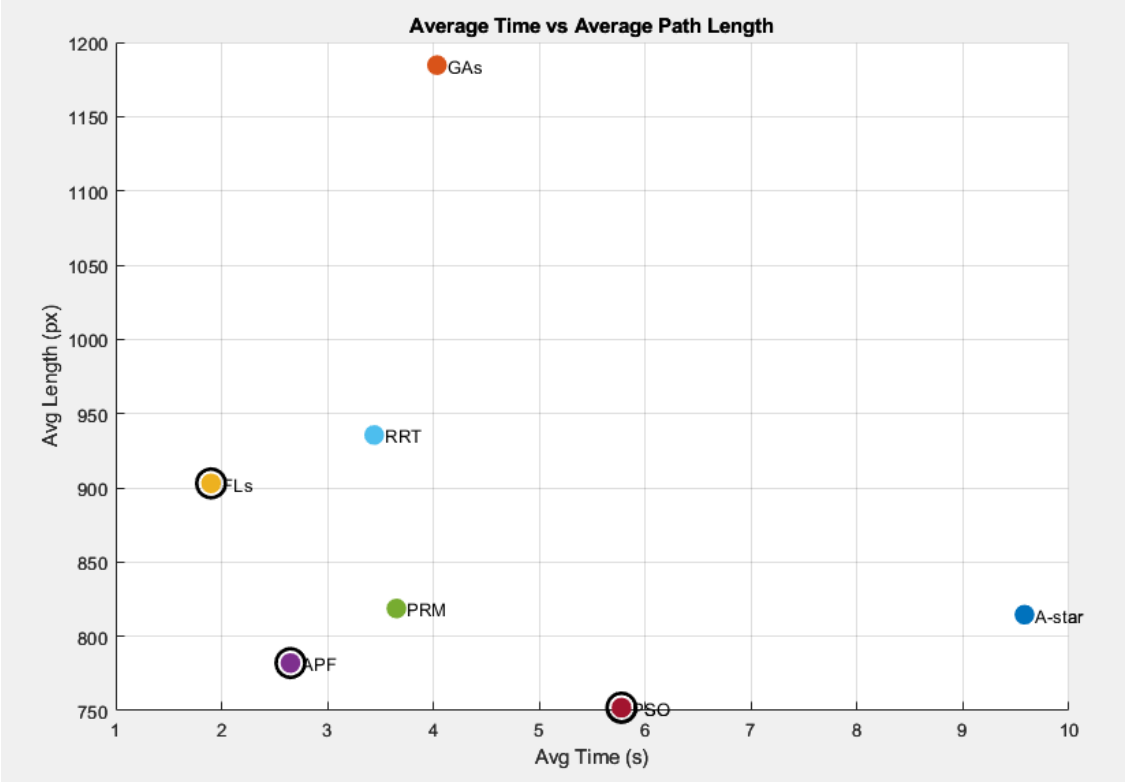
4.5 Results and Analysis

The evaluation of algorithms for mobile robot guidance revealed distinct strengths and weaknesses, depending on the scenarios. Table 4.5 summarizes the raw results, while Table 4.6 provides aggregated performance across all environments.

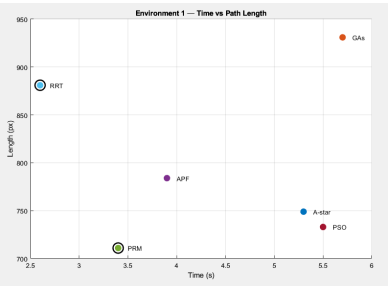
Table 4.5: Results of path planning algorithms across all environments

		Environments				
Algorithm	**Criteria**	1	2	3	4	5
A-star	Time	5.3 s	7.7 s	10.6 s	12.0 s	12.3 s
	Length	749 px	854 px	899 px	702 px	869 px
GAs	Time	5.7 s	2.4 s	—	4.0 s	—
	Length	931 px	1710 px	—	913 px	—
FLs	Time	—	—	—	1.9 s	—
	Length	—	—	—	903 px	—
APF	Time	3.9 s	—	—	1.4 s	—
	Length	784 px	—	—	780 px	—
PRM	Time	3.4 s	3.7 s	—	3.3 s	4.2 s
	Length	711 px	902 px	—	682 px	980 px
RRT	Time	2.6 s	2.9 s	6.4 s	2.4 s	2.9 s
	Length	881 px	1000 px	1057 px	758 px	982 px
PSO	Time	5.5 s	5.5 s	—	5.2 s	6.9 s
	Length	733 px	836 px	—	717 px	722 px

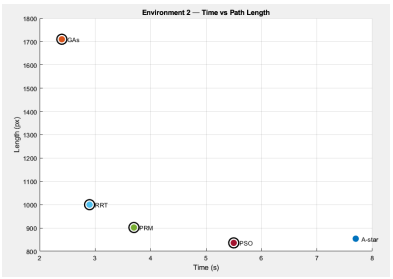
— indicates failure to find a path. s -> seconds, px -> pixels



(a) Average Time–Length Performance

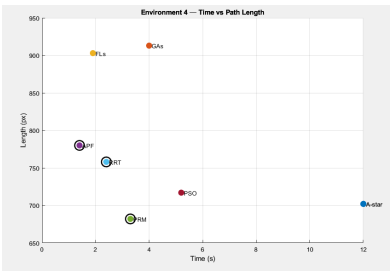


(b) Environment 1

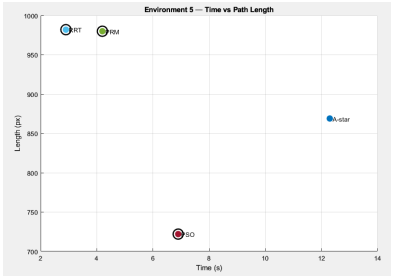


(c) Environment 2

(d) Environment 3



(e) Environment 4



(f) Environment 5

Figure 4.16: Time–Length diagrams for all path planning algorithms.

Note: All values in Table 4.5 correspond to the average of ten independent runs for each algorithm in each environment. This repeated-run scheme minimizes stochastic variability—especially important for sampling-based and population-based planners—ensuring that the reported results reflect stable and representative

Table 4.6: Aggregate performance across all environments (based on Table 4.5).

Algorithm	**Avg. Time**	**Avg. Path Length**	**Success Rate**
A-star	9.6 s	814.6 px	100%
GAs	4.0 s	1184.7 px	60%
FLs	1.9 s	903 px	20%
APF	2.6 s	782 px	40%
PRM	3.7 s	855 px	100%
RRT	3.4 s	935.6 px	100%
PSO	5.8 s	752 px	80%

performance rather than single-run anomalies. Cells marked with "—" indicate that the algorithm failed to generate a valid path in all ten trials for that environment.

The corresponding time–length diagrams are provided in Fig. 4.16, where panel (a) shows the aggregated (average) performance, and panels (b–f) illustrate environment-specific results.

4.5.1 Discussion of Results

A-star consistently produced valid paths in all environments, but its computation time increased significantly with complexity, reflecting its exhaustive search nature.

GAs worked well in open settings but failed in constrained environments; such as maze-like environments, due to insufficient population size. This highlights the need for parameter adjustment to improve robustness.

The FLs succeeded only in one case, demonstrating fast computation but poor generalization across various obstacle configurations.

APF found feasible paths in two environments with short computation times but struggled with local minima, a known limitation of potential-field methods.

PRM generally produced short paths and maintained a high success rate, except in narrow passages where the sampling density was inadequate.

RRT successfully solved all test cases and excelled in complex spaces. although the generated paths were longer, this reflected its bias towards feasibility over optimality.

PSO achieved near-optimal path lengths in most environments and had a success rate of 80%. Although slightly slower than PRM or RRT, it balanced exploration and exploitation effectively. For this reason, PSO was selected as the primary method for further investigation.

In summary, the performance of the algorithm was highly dependent on the environmental structure. We next quantify these differences using ANOVA and Tukey HSD on execution times.

4.5.2 Statistical Analysis (ANOVA)

ANOVA is a statistical method used to test whether the means of multiple groups are significantly different [111]. ANOVA tries to determine whether the mean differences between two or more groups are merely a coincidence or whether there is a truly significant difference.

To statistically assess whether there are significant differences in execution times between different path planning algorithms, a one-way analysis of variance, namely **ANOVA** test, was conducted here. This test evaluates whether the mean execution times across multiple groups, namely algorithms, are significantly different from one another.

Dataset: To ensure a fair and systematic comparison, it was essential to assess all candidate algorithms on the identical dataset. From our initial collection, four algorithms: **A-star**, **RRT**, **PRM**, and **PSO**, exhibited commendable performance, showing complete or nearly complete outcomes in all five trial environments. This preliminary evaluation indicated that these four algorithms are the most suitable for a detailed study. Therefore, our thorough analysis concentrated on this chosen subset, excluding other algorithms such as GA, APF, and FLs from the final comparison tests.

Table 4.7: Execution times across 5 environments, illustrating the relative speed differences between algorithms. Notably, RRT consistently requires less computation time compared to A-star and PSO, which highlights its suitability for time-critical applications.

Algorithm	Env 1	Env 2	Env 3	Env 4	Env 5
A-star	5.3	7.7	10.6	12.0	12.3
RRT	2.6	2.9	6.4	2.4	2.9
PRM	3.4	3.7	—	3.3	4.2
PSO	5.5	5.5	—	5.2	6.9

Preprocessing: The missing values for Environment 3 in PRM and PSO were excluded pairwise. ANOVA was applied only to environments where all four algorithms had valid execution time data: Environments 1, 2, 4, and 5.

Hypotheses:

- H_0: There is no significant difference between the mean execution times of the algorithms ($\mu_1 = \mu_2 = \mu_3 = \mu_4$).
- H_A: At least one algorithm has a significantly different mean execution time.

Test Results: The ANOVA test yielded the following results:

- **F-statistic:** 11.12
- **p-value:** 0.000885

Since the p-value is significantly less than 0.05, the null hypothesis is rejected. This confirms that execution times differ significantly between algorithms. All results can be viewed in Table 4.7.

Effect Size: In addition to statistical significance, the strength of the observed differences was evaluated using **eta squared** (η^2), which quantifies the proportion of variance in execution times explained by the choice of algorithm.

Effect Size

While statistical significance (p-values) indicates whether differences exist between groups, it does not convey the magnitude of these differences. Effect size metrics complement this by quantifying how strong or practically meaningful the observed differences are. In ANOVA contexts, measures such as eta squared (η^2) and Cohen's f are commonly used. According to Cohen's conventions [142], η^2 values of 0.01, 0.06, and 0.14 correspond to small, medium, and large effects, respectively. Thus, the obtained $\eta^2 = 0.68$ and $f = 1.46$ clearly indicate a large effect, meaning that algorithm choice strongly influences execution time.

The computed effect size was:

- $\eta^2 = 0.68$ (large effect, according to Cohen's conventions).

This indicates that approximately 68% of the variability in execution times can be attributed to differences between algorithms, confirming that the impact of algorithm choice on execution time is not only statistically significant but also practically meaningful.

For additional interpretability, Cohen's **f** effect size measure **f** was also derived:

- $f = 1.46$ (large effect).

Both indices reinforce that algorithm selection has a substantial influence on performance beyond what can be attributed to random variation.

Post-hoc Analysis (Tukey HSD): To identify which algorithms differ significantly, the Tukey's Honest Significant Difference (HSD) test was performed, and the results can be seen in Table 4.8.

It is important to note that the outcomes of the statistical analysis are consistent with the descriptive results presented earlier in Tables 4.5 and 4.6. Specifically, A-star shows significantly higher execution times compared to RRT and PRM, which aligns with the raw averages. Meanwhile, no significant differences were detected between RRT, PRM, and PSO, which is also reflected in their comparable mean values. Thus, the ANOVA and Tukey HSD results confirm that the observed performance gaps are not only visible in descriptive statistics but are also statistically robust.

Interpretation: The results indicate that **A-star** is significantly slower than both **RRT** and **PRM**. However, its difference from **PSO** is not statistically significant. Among RRT, PRM, and PSO, no significant

Table 4.8: Tukey HSD pairwise comparison results. The table shows that A-star is significantly slower than both RRT and PRM, confirming that algorithm selection has a substantial effect on execution time.

Group 1	Group 2	Mean Diff.	p-value	CI Low	CI High	Significant
A-star	RRT	7.88	0.001	3.48	12.27	Yes
A-star	PRM	6.10	0.006	1.71	10.49	Yes
A-star	PSO	3.90	0.104	-0.49	8.29	No
RRT	PRM	-1.78	0.595	-6.17	2.62	No
RRT	PSO	-3.98	0.100	-8.37	0.42	No
PRM	PSO	-2.20	0.474	-6.60	2.20	No

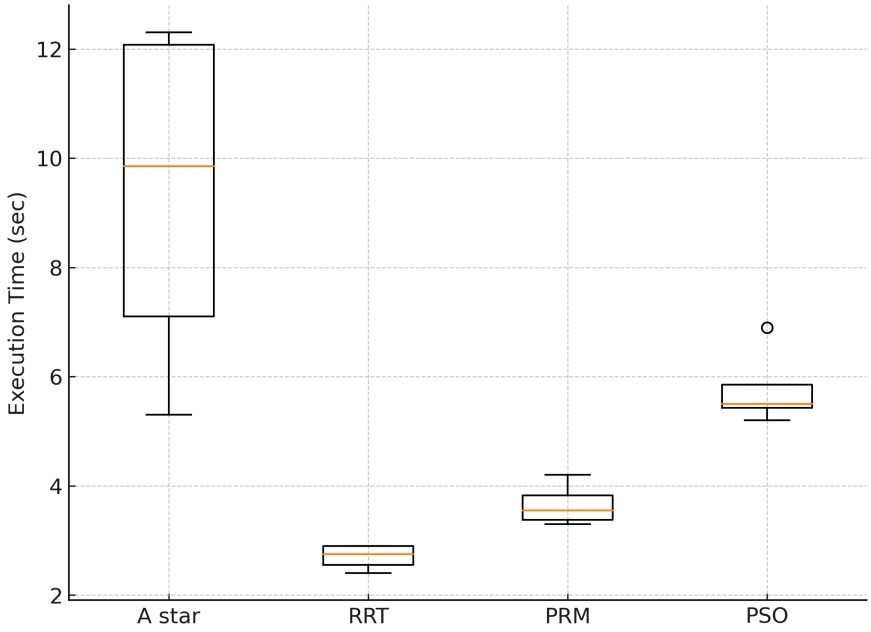


Figure 4.17: Execution time boxplots across algorithms (Env. 1, 2, 4, 5). A-star shows higher median and spread; RRT and PRM are lower and more consistent.

differences were found. This suggests that while A* ensures reliability and accuracy, it comes at the cost of execution time. In contrast, RRT and PRM consistently offer faster solutions.

About Boxplots

A boxplot is a graphical summary that illustrates the distribution of data based on median, quartiles, and potential outliers [142]. It helps to visually compare central tendency and variability across groups. In our data (Envs. 1,2,4,5), A-star shows higher median and variability; RRT/PRM are lower and more consistent (Fig. 4.17). Such visual evidence complements the statistical findings and strengthens the overall interpretation.

These statistical findings (Fig. 4.17) reinforce the empirical results presented earlier. RRT and PRM are better suited for time-critical applications, while A-star prioritizes path optimality at the expense of

longer computation time. PSO shows intermediate behavior, providing a balance between solution quality and execution speed.

4.5.3 Single-objective Path Planning

Most of the algorithms presented in this chapter focus primarily on minimizing the path length. However, real-world navigation involves multiple and often competing objectives, such as:

- Minimizing total path length,
- Maximizing smoothness and continuity,
- Reducing collision risk and ensuring safety margins,
- Conserving energy,
- Balancing time efficiency with computational complexity.

Many implementations use a scalarized surrogate objective with fixed weights (Eq. 4.32), which eases tuning but may obscure trade-offs:

$$\min_{P} \left(\alpha_1 \cdot \frac{\text{Length}(P)}{L_{\max}} + \alpha_2 \cdot \frac{\text{Smoothness}(P) - S_{\min}}{S_{\max} - S_{\min}} + \alpha_3 \cdot \frac{\text{Risk}(P)}{R_{\max}} + \alpha_4 \cdot \frac{\text{Energy}(P)}{E_{\max}} \right), \quad (4.32)$$

where α_i are fixed nonnegative weights. Eq. 4.32 is therefore not a genuine multi-objective path planning (MOPP) problem but a single-objective surrogate used for simplicity. It enables trade-off studies but inherits the limitations of linear weighting when combining incommensurable criteria.

4.5.4 Multi-objective Path Planning

A true MOPP formulation simultaneously optimizes several conflicting objectives without collapsing them into a single aggregate. Formally:

$$\min_{P} F(P) = \big(f_1(P), f_2(P), \ldots, f_m(P)\big), \quad (4.33)$$

subject to feasibility constraints in Eqs. 4.4–4.7.

The solution is not a single path but a Pareto-optimal set:

$$\mathcal{P}^* = \{P \mid \nexists Q : f_j(Q) \leq f_j(P)\, \forall j,\ f_k(Q) < f_k(P) \text{ for some } k\}. \quad (4.34)$$

This Pareto set captures the inherent trade-offs between objectives: for example, one path may minimize length while another minimizes risk. Such formulations avoid the pitfalls of scalarization, which may distort optimization results when objectives differ in scale or semantics [178, 173].

Pareto-based approaches; NSGA-II planners, risk-aware RRTs, maintain sets of alternatives for context-aware selection [209, 138]. These approaches allow a robot or supervisory decision module to select among feasible solutions according to mission context, namely, prioritizing safety in hazardous zones versus energy efficiency in long-range missions.

In summary, adopting multi-objective formulations provides a more principled foundation for robotic navigation. Instead of a fixed "shortest path," future planners can balance safety, efficiency, and mission constraints through Pareto-optimal alternatives, thus achieving context-aware navigation.

4.6 Summary

This chapter provides a detailed analysis of established path planning algorithms, seeking to illuminate their advantages and limitations. The study begins with a foundational overview of these algorithms and progresses to a comparative assessment to measure their suitability for various environments. Given the vast array of available methodologies, this chapter focuses exclusively on the notable algorithms identified through a literature review. In addition, the criteria and processes used for their selection are clarified.

The next chapter builds upon these foundations by presenting the proposed methodology. It introduces a decision-making system that integrates learning, dynamic prioritization, and real-time damage estimation to enable mission-resilient autonomy.

Chapter 5

Methodology

This chapter details the methodology used to integrate a metaheuristic path planner with learning-based obstacle assessment. We (i) formalize the decision problem (avoid or traverse), (ii) describe data generation and feature processing, (iii) present PCA and clustering to obtain obstacle categories, (iv) train a regression model to estimate damage, and (v) couple these elements with PSO for planning.

The chapter is organized into eight sections that follow the data-to-decision pipeline. Fig. 5.1 summarizes component interactions.

1. **Research Design**: We evaluate an integrated navigation system that couples a PSO-based planner with a learned damage estimator. Section 5.2 specifies hypotheses, variables, and the evaluation protocol.

2. **Deciding to Avoid Obstacles**: The robot's decision to navigate away from obstacles depends more on assessing potential harm than on completely evading them. It analyzes each obstacle, taking into account aspects such as the severity of the potential impact. When the estimated damage is low, the robot may keep the current route instead of avoiding it. We test the effect of this policy on path cost and safety in "Experimental Results and Analysis Chapter".

3. **Collision Records**: Records of collisions contain details about past encounters between the robot and obstacles. These records aid the robot in learning from prior interactions, allowing it to assess the severity of possible impacts. Previous interaction records are utilized solely for training and validating the estimator; their influence on the quality of decisions is assessed subsequently through controlled simulations.

4. **Principal Component Analysis (PCA)**:PCA serves as a dimensionality reduction method aimed at simplifying intricate datasets while retaining essential information. Within path planning, PCA assists in determining the most crucial elements that affect navigation choices. By decreasing the number of variables, PCA boosts computational efficiency and enhances the robot's capacity to handle data during damage estimation. This method helps to identify the types of obstacles in raw data.

5. **Elbow Method**: The elbow method is a preliminary strategy in clustering analysis aimed at identifying the ideal number of clusters within a dataset. This involves graphing the internal cluster variance versus the number of clusters and pinpointing where the reduction in the variance rate diminishes,

creating an "elbow" shape in the graph. Within path-planning studies, this approach can classify obstacles or navigation patterns according to their similarities. Choosing the best number of clusters enables a robot to categorize obstacles more efficiently, enhancing its decision-making capabilities. We use the elbow heuristic to propose candidate k values; the final k is validated against downstream decision performance (section 5.6).

6. **Decision-Making Algorithm**: This algorithm achieves two things; path planning and decision-making. It evaluates whether the robot should avoid or traverse an obstacle by considering the current data during the journey. It takes into account elements such as previous collision incidents, potential damage, and travel efficiency.

7. **Learning**: In this section, 2 different ML techniques are used; clustering and regression. *Clustering* is a technique in data analysis that groups together similar items according to common attributes. In the realm of robotic navigation, this method aids in grouping obstacles by their severity of impact, material characteristics, or how frequently they are encountered. Such classification supports the decision-making algorithm in adopting varied navigation strategies, depending on the type of obstacle. We hypothesize that such grouping helps decision policies; we assess this in ablation studies.

   The second technique **is Multiple Linear Regression** , which is a statistical technique used to model the relationship between one dependent variable and two or more independent variables. It estimates how changes in the independent variables are associated with changes in the dependent variable by fitting a linear equation to the observed data.

8. **Summary**: The summary offers a general overview of the research, emphasizing important results and contributions. It describes how the new method enhances conventional path-planning by incorporating a dynamic decision-making mechanism. This work advances autonomous robotic systems by providing a more intelligent and adaptable navigation strategy.

Lastly, the *AnticipatorySim* MATLAB simulation platform was selected during the *third level tests* because of its high flexibility and compatibility with custom MATLAB functions [183, 174]. It provided good integration with learning-based modules and enabled rapid prototyping of decision-making functions.

5.1 Obstacle Decision and Learning-based Coordination

We frame obstacle handling as a two-action decision; avoid, traverse, under task-specific cost trade-offs. Rather than treating every obstacle uniformly, robots must evaluate whether circumventing or traversing an obstacle yields the best outcome under the given constraints. Therefore, before introducing the *Research Design*, this section defines the obstructive decision problem that underpins the proposed learning-based coordination framework.

By integrating predictive modeling with rule-based reasoning, the framework enables real-time decision making that balances path efficiency against potential risks. This link between obstacle evaluation, learning-based coordination, and overall mission performance constitutes the foundation of this thesis.

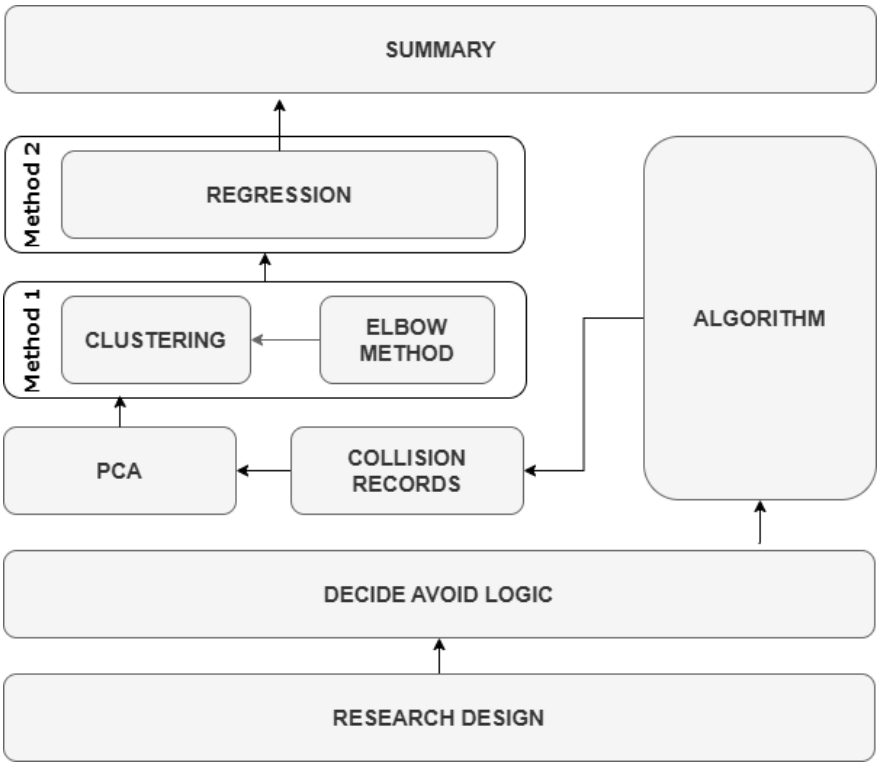


Figure 5.1: Methodology flow.

5.1.1 Key Definitions

Definition 5.1 (Robot)**.** A robot R is an autonomous agent that navigates in a known or partially known environment $\mathcal{W} \subset \mathbb{R}^2$ by following a path P from the start position q_{start} to the goal position q_{goal}.

Definition 5.2 (Path)**.** A path P is a finite sequence of points $P = \{q_0, q_1, ..., q_n\}$ where each $q_i \in \mathcal{C}_{\text{free}} \subseteq \mathcal{W}$ denotes the collision-free configuration space, and $\mathcal{C}_{\text{free}}$.

Definition 5.3 (Obstacle)**.** An obstacle $o_i \in \mathcal{O}$ is a region in the environment $\mathcal{W}$ such that $o_i \cap P \neq \emptyset$ implies a potential conflict between the robot and the environment, requiring a decision on whether to avoid or traverse it.

Definition 5.4 (Path Cost)**.** The path cost function $J(P)$ evaluates the overall effort to traverse a path P based on criteria such as travel time, distance, or energy use. Formally, $J : \mathcal{P} \to \mathbb{R}_{\geq 0}$ maps each path to a scalar cost value.

Definition 5.5 (Damage Cost)**.** The damage cost $D(o_i)$ estimates the potential harm to the robot or mission if the obstacle o_i is traversed rather than avoided. It is calculated by a predictive model $\mathcal{M}$ trained on collision-related features derived from prior data.

Definition 5.6 (Decision Rule)**.** Given an obstacle o_i encountered along the current path P, we compare the marginal avoid cost against the predicted traversal damage:

$$C(A_1) = J(P_{\text{new}}) - J(P), \qquad C(A_2) = D(o_i), \tag{5.1}$$

where A_1 denotes replanning around o_i and A_2 denotes *traversing* it. The selected action is

$$A^* = \begin{cases} A_1, & \text{if } C(A_1) < C(A_2) \\ A_2, & \text{otherwise.} \end{cases} \tag{5.2}$$

5.2 Research Design

This study adopts a simulation-driven experimental approach to investigate a hybrid navigation system that integrates shortest-path optimization with learning-based obstacle assessment. *The central objective is to validate whether the proposed model enables a mobile robot to make dynamic decisions on whether to avoid or traverse obstacles encountered, balancing predicted damage against the overall cost of the path.*

The research process is structured into three phases:

1. **Model Implementation**: The proposed navigation framework combines a metaheuristic path planner with a predictive model trained to estimate the potential damage of obstacles. The decision rule (Eq.5.1) evaluates alternative actions using the weighted cost function:

$$C = \alpha \cdot \frac{T}{T_{\max}} + \beta \cdot \frac{E}{E_{\max}} + \gamma \cdot \frac{D}{D_{\max}} \tag{5.3}$$

   where T is the travel time, E is the energy usage, and D is the predicted damage. All terms are normalized to a common scale $[0, 1]$ using maximum plausible values $T_{\max}$, $E_{\max}$, and $D_{\max}$ to ensure unit consistency. The coefficients α, β, γ allow for flexible prioritization of mission-specific objectives.

   The weighting parameters (α, β, γ) in the function were determined based on evaluation and theoretical considerations. Specifically, α emphasizes the minimization of path length, γ penalizes collisions with obstacles, and β accounts for energy and time efficiency. The initial values will be selected through pilot simulations and tuned to balance safety with performance. The chosen weights reflect a trade-off: excessive emphasis on α may cause unsafe shortcuts, while high values of β or γ may lead to overly conservative paths. This balanced configuration ensures both feasibility and adaptability in diverse scenarios.

2. **Simulation Experiments**: The model is tested in a MATLAB-based environment adapted from the *AnticipatorySim* simulation platform. Scenarios are designed with varying obstacle properties to capture a range of navigation challenges. For each scenario, performance metrics such as total path cost, the number of collisions, and travel efficiency are recorded.

3. **Comparative Evaluation**: Experimental results are compared with established algorithms that focus solely on path optimization. The comparison highlights whether the integrated model achieves superior trade-offs in environments with different levels of risk and constraint.

Although the experiments are carried out in simulation, *their design incorporates elements of real-world uncertainty, including perception noise, actuator variability, and resource limitations*. These considerations provide insight into how the proposed framework may perform in physical deployments such as autonomous exploration or robotic emergency response.

5.2.1 Comparison of Alternative Algorithms

To validate the effectiveness of the proposed PSO-based approach, seven alternative algorithms from Chapter 4 were selected for comparison: A-star, RRT, PRM, GA, APF, Fuzzy Logic, and PSO. Each was implemented under identical simulation conditions. The evaluation considered not only path length and computation time but also obstacle handling behavior and mission success rate.

Table 4.6 summarizes the averaged results of 10 test cases in environments with varying obstacle densities and damage potentials. All baseline algorithms were parameterized following widely accepted configurations in the literature.

In Table 4.6, A-star and RRT achieved competitive path lengths but expended additional length and time on excessive avoidance maneuvers, as obstacle severity was not evaluated. PRM and Fuzzy Logic demonstrated unstable performance in densely cluttered environments, while APF frequently failed due to local minima. The GAs offered shorter paths in open spaces but required longer computation times.

In contrast, PSO consistently balanced efficiency and safety. Beyond producing competitive path lengths, its ability to incorporate damage-aware decision rules allows the robot to traverse low-risk obstacles when advantageous. This resulted in better overall success rates in risk-prone environments.

For these reasons, PSO was selected as the core optimization engine for integration with machine learning components in subsequent chapters.

5.3 Decide to Avoid Obstacles

The first objective of this study is to address the limitations of conventional path planning methods, which enforce hard collision constraints; even when obstacles pose negligible risk [94]. Such behavior often results in unnecessarily long paths and increased energy consumption. In contrast, the proposed approach improves the flexibility of robotic navigation by introducing a decision-making mechanism that weighs the expected damage from obstacle interaction against the additional cost of avoidance. By considering the severity of potential harm in real time, the robot can dynamically decide whether to adhere to the original path or to replan [47]. *The ultimate goal is to achieve autonomous navigation that is both energy efficient and risk-aware*.

5.3.1 Decision Mechanism

The proposed system follows a cascading sequence of steps:

1. Compute the shortest path in the environment.

2. When an obstacle is encountered, estimate the potential damage using a predictive model trained on past experiences.
    - If the estimated damage exceeds an acceptable threshold, replan the path and calculate the additional travel costs.
    - If the damage is below the threshold, continue along the current path and traverse the obstacle.

This formulation moves beyond the deterministic paradigm of always avoiding obstacles. Instead, the robot selects between two competing actions, avoidance versus traversing, based on a comparative evaluation of *additional length/time required* versus *expected damage cost*. This balance allows robots to handle obstacles with graded responses rather than binary avoidance.

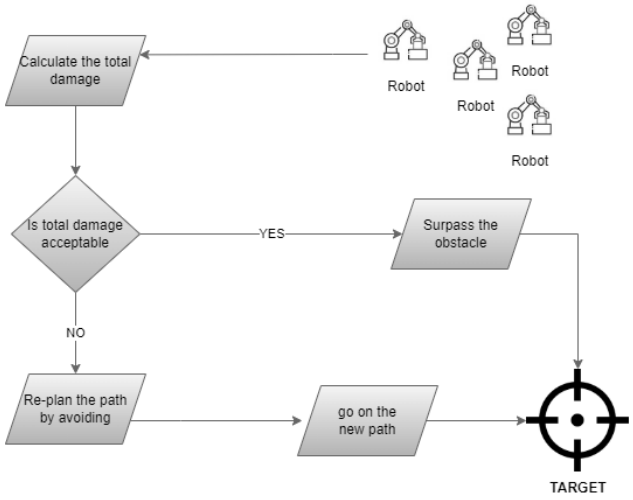


Figure 5.2: General workflow of the proposed system.

5.3.2 Path Optimization via PSO

PSO was selected as the underlying optimizer for path generation. As shown in earlier evaluations (*see Table 4.6*), PSO produced competitive path lengths and proved effective in dynamic settings. Its population-based search mechanism makes it suitable for environments with varying obstacle densities, as it iteratively explores and refines candidate paths toward high-quality solutions.

5.3.3 Damage Estimation Model

A key element of the framework is the prediction of potential damage when an obstacle is encountered. Since obstacle types, robot characteristics, and environmental layouts differ between scenarios, training data are generated to represent diverse conditions. These records include obstacle severity scores, robot profiles, and simulated collision outcomes. The learning model is then trained on these data to estimate the damage cost $D(o_i)$ associated with traversing a specific obstacle.

At runtime, when the robot faces an obstacle, the predictive model outputs an estimated damage value. If this value falls below the predefined threshold, the robot continues on its original trajectory. Otherwise, the system triggers a replanning step to generate an alternative path.

In summary, this section introduced a novel navigation mechanism that jointly evaluates *path cost* and *predicted damage cost*. Unlike traditional approaches that avoid all obstacles by default, the proposed method allows mobile robots to take calculated risks, thereby improving efficiency while maintaining safety. The cascading flow of this decision process is illustrated in Fig. 5.2.

5.4 Collision Records

This study uses two established real-world datasets to derive a synthetic collision dataset for training and evaluation:

- **M2DGR**: Multi-sensor and multi-scenario SLAM dataset [235].
- **The Rosario Dataset**: Multisensor data collected in agricultural environments [76].

Although these datasets do not provide explicit collision annotations, they contain synchronized multi-modal sensor streams (*IMU, GPS-RTK, wheel encoders, and stereo cameras*) that enable the identification of anomalous patterns. By analyzing deviations in vibration signals, velocity fluctuations, and perception inconsistencies, potential collision-like events were inferred.

From these events, feature vectors were systematically constructed to approximate realistic robot–obstacle interactions. The extracted features include *robot velocity, impact angle, obstacle geometry and material properties, estimated impact force, vibration levels, visual distortions, and sensor anomalies*. Table 5.1 illustrates an excerpt from the generated records.

All feature values were computed in MATLAB by applying physical approximations and scenario-based assumptions grounded in the robotics literature. In this way, the dataset captures various damage scenarios involving different types of obstacles, such as static walls, uneven terrain, and dynamic agents, while preserving physical plausibility.

Table 5.1: Records of derived collision data

Metric	Row 1	Row 2	Row 3	Row 4
Robot Velocity (m/s)	14.23	13.20	13.16	14.37
Impact Angle (rad)	1.71	1.78	2.36	1.95
Obstacle Surface Hardness	2.43	2.14	2.67	2.50
Obstacle Height (cm)	15.6	11.2	18.6	16.8
Material Weight (kg)	127	100	101	113
Impact Force (kN)	2.8	2.65	2.8	3.85
Post-Impact Vibration (g)	0.28	0.26	0.30	0.24
Visual Blur Index	5.64	4.38	5.68	7.80
Lidar Anomaly	1.04	1.05	1.03	0.86
Sensor Noise Level	1065	1050	1185	1480
Obstacle Type	1	1	3	2

In conclusion, the constructed dataset reflects collision dynamics under realistic assumptions without resorting to arbitrary values. By incorporating variation in obstacle features, robot responses, and sensor

anomalies, it enables the development and validation of predictive models for damage-aware path planning. As seen in Table 5.1, the dataset captures the geometric and physical characteristics of obstacles, providing a robust basis for learning-based decision models.

5.5 Principal Component Analysis

Building on the anomaly-inferred records derived from M2DGR and the Rosario dataset (*Section 5.4*), we assembled a unified feature matrix X containing sensor- and obstacle-related variables, such as robot velocity, surface hardness, obstacle geometry, and visual and lidar anomaly indicators. The curated dataset was exported as a CSV file for use in the learning pipeline.

PCA is a widely used dimensionality-reduction technique that transforms possibly correlated variables into a smaller set of uncorrelated linear variables, known as principal components, while retaining most of the variance [53]. This is particularly helpful with high-dimensional sensor streams, where collinearity and noise can obscure structure. PCA projects the data onto a new coordinate system such that the first component captures the maximum variance, and each subsequent component captures the next highest variance in an orthogonal direction.

Mathematically, PCA is obtained via the eigenvalue decomposition of the covariance matrix:

$$\Sigma \;=\; \frac{1}{n-1}\,X^{\top}X \;=\; V\Lambda V^{\top}, \tag{5.4}$$

where X is the zero-mean data matrix (rows: observations; columns: features), Σ is the covariance matrix, V contains the eigenvectors (*principal axes*), and Λ is the diagonal matrix of eigenvalues that quantifies variance along each component [53].

In MATLAB, PCA was applied using:

```
[coeff, score, latent, tsquared, explained] = pca(X);
```

Here, `score` contains the low-dimensional representation, `coeff` the loading vectors, and `explained` the per-component variance ratios.

In our pipeline, PCA served two purposes: (i) reducing redundancy to improve downstream learning and (ii) exposing structure in the data that relates obstacle properties to outcome-relevant behavior. Consistent with the clustering stage, the retention of three components preserved 92–96% of the variance, allowing a compact but expressive representation for later stages. The projection revealed separable patterns that aligned with obstacle semantics; in particular, continuous sensory signatures mapped onto more distinguishable categories such as *liquid*, *peak*, and *mound*. These groupings were subsequently exploited by the clustering module and the decision mechanism to differentiate when traversing is acceptable versus when avoidance is necessary.

As shown in Fig. 5.3, the PCA projection provides a concise view of similarity across records: points that are closer in the principal component space tend to share similar physical and sensor profiles, and thus similar expected impacts on navigation cost and damage. This compact representation supports robust learning while maintaining interpretability within the methodology.

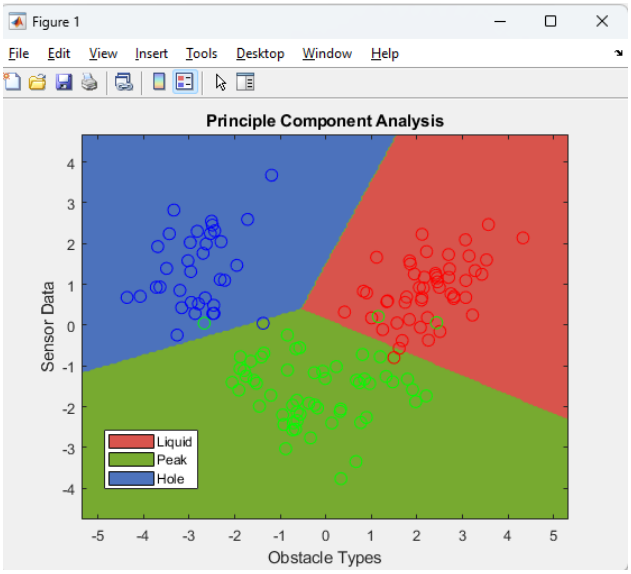


Figure 5.3: Obstacle types visualized after dimensionality reduction via PCA

Unlike standard PCA usage, here the projection not only reduced redundancy but also revealed semantically meaningful obstacle categories (liquid, peak, mound), which were directly exploited in the decision-making pipeline. This highlights the originality of the method by linking dimensionality reduction with task-specific navigation semantics.

5.6 Elbow Method

In order to identify natural groupings within the collision records, the elbow method was used. This heuristic provides a practical way to estimate the optimal number of clusters (k) by evaluating how the quality of the clustering improves as the number of clusters increases. As seen in Table 5.1 and the PCA projection (*Section 5.5*), the dataset contains a wide variety of obstacle impact characteristics, making an unsupervised grouping step essential for downstream learning.

The underlying principle of the Elbow Method is that increasing k will generally reduce the variance within the cluster, since each additional cluster can more closely capture the local structure [123]. However, after a certain point, the improvement becomes marginal, as additional clusters merely subdivide existing groups without adding significant explanatory value. This trade-off produces the characteristic "elbow" shape in the evaluation curve, where the slope of improvement noticeably decreases.

Formally, the method involves computing the total within-cluster sum of squares (WCSS) for each candidate k:

$$\text{WCSS}(k) = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2 \tag{5.5}$$

where C_i is the cluster i and μ_i is its centroid. The optimal k is the point at which the rate of decrease in WCSS rapidly levels off, indicating that additional clusters contribute little new information.

The elbow appears near $k \in \{4, 5\}$; we select $k = 5$ based on Fig. 5.4. The curve flattens noticeably after five clusters, implying that five groups adequately capture the variation in collision records without unnecessary fragmentation. These clusters align with the obstacle categories observed in the PCA projection, strengthening their interpretability.

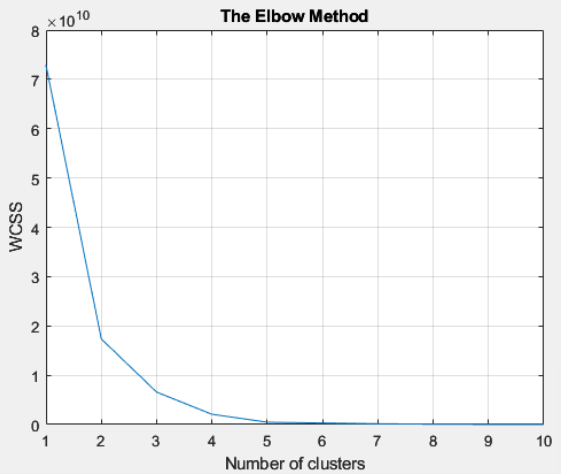


Figure 5.4: Elbow curve showing the optimal number of clusters at $k = 5$.

This clustering result is consistent with the separation observed in Fig. 5.3, where the dimensionality-reduced space suggested distinct categories of collision dynamics. By confirming the cluster count with the Elbow Method, we ensure that subsequent learning-based coordination models are built upon stable and interpretable group structures.

In contrast to typical clustering setups, the elbow point at k=5 was not treated as an abstract statistical optimum, but was validated against PCA-derived obstacle semantics. This dual confirmation establishes a unique methodological link between dimensionality reduction and safety-aware clustering in robotic navigation.

5.7 Decision-Making Algorithm

The proposed Decision Making Algorithm improves the efficiency and adaptability of mobile robots by combining a metaheuristic path planner with a learning-based damage assessment model. Unlike traditional path planning strategies that treat all obstacles equally hazardous, this approach assesses obstacles in real time and determines whether avoidance or traversal yields the lowest overall cost. Integration of PSO with predictive modeling enables a balance between path efficiency and survivability.

5.7.1 Initialization

- Train a predictive model using processed features to estimate obstacle damage.
- Initialize a swarm of particles, each representing a candidate route.

- Define key parameters: particle count, maximum iterations, inertia weight, and learning coefficients.
- Set a maximum permissible damage threshold that is consistent with mission requirements.

5.7.2 Path Evaluation

For each particle (path candidate):

- Compute path length and approximate energy use.
- Detect obstacles encountered along the route.
- For each obstacle:
  - Extract relevant features, such as velocity, approach angle, and obstacle type.
  - Use the predictive model to estimate potential damage.
  - Determine whether the obstacle is traversable based on the threshold.

5.7.3 Decision Mechanism

The action choice follows a simple rule:

- **If the predicted damage** is $>$ the threshold, avoid it.
- **Else**: **Traverse**; namely, continue on the existing path.

5.7.4 Formulation

The fitness of a candidate path P_i is defined as:

$$J(P_i) = \alpha \cdot \text{Length}(P_i) + \beta \cdot \text{Damage}(P_i) \tag{5.6}$$

where:

- $\text{Length}(P_i)$ is the total path length,
- $\text{Damage}(P_i)$ is the cumulative predicted damage,
- α and β are weights that balance efficiency and safety.

PSO updates each particle according to the standard formulation [95]:

$$v_i^{t+1} = w \cdot v_i^t + c_1 \cdot r_1 \cdot (p_i - x_i^t) + c_2 \cdot r_2 \cdot (g - x_i^t) \tag{5.7}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{5.8}$$

where v_i^t is the velocity, x_i^t the position, p_i the personal best, g the global best, and w, c_1, and c_2 the inertia and learning coefficients.

To enforce safety, paths that violate the maximum damage constraint must be identified.

$$\text{Damage}(P_i) \leq D_{\text{max}} \tag{5.9}$$

are discarded. D_{max} is set based on mission safety constraints; we sweep values and report sensitivities. Please see Algorithm C.14 in Appendix C for the full pseudocode.

5.7.5 PSO Iteration and Convergence

- Update swarm positions based on individual and global bests.
- Recalculate costs using the combined path–damage function.
- Continue until the iteration limit is reached or the improvement falls below a threshold.

5.7.6 Output

The final output is the path with:

- Minimum overall cost,
- Acceptable energy consumption,
- Damage values are within the safety limit.

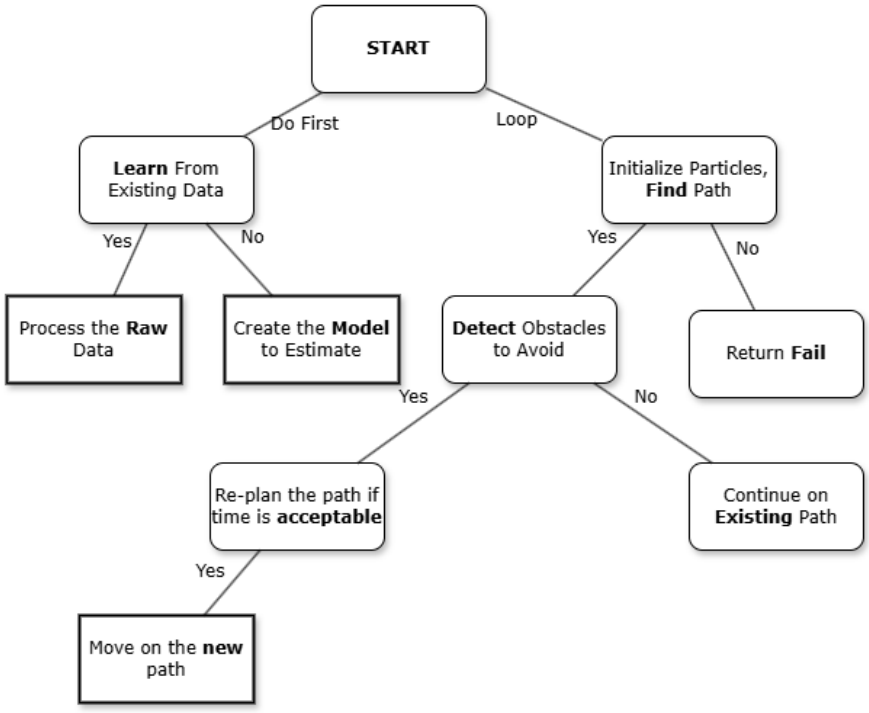


Figure 5.5: Flow of the integrated PSO and damage-aware decision model.

5.7.7 Advantages

- Reduces unnecessary avoidance, thereby improving energy efficiency.
- Integrates real-time obstacle damage prediction into path planning.
- It provides adaptability in dynamic environments where not all obstacles are equally critical.

This algorithm is particularly relevant for autonomous vehicles and mobile robots operating in uncertain environments. By coupling PSO with predictive assessment, the system addresses both classical efficiency criteria and safety considerations within a unified decision framework.

The novelty of this approach lies in the extension of PSO beyond traditional path optimization by embedding damage-aware predictions into the cost function. This integration transforms the PSO into a hybrid decision mechanism that balances efficiency with structural safety, specifically tailored for autonomous navigation under uncertain obstacle conditions.

5.8 Learning

This section describes the learning-based components of the proposed framework. Two complementary models were developed: an unsupervised clustering model for obstacle categorization and a supervised regression model to predict potential collision damage. Together, these models provide the foundation for adaptive decision-making.

5.8.1 Clustering Model: K-Means

To identify natural groupings among obstacles, K-Means clustering was applied to the PCA-reduced feature set. The objective was to categorize obstacles with similar structural and risk-related properties.

Input Features :

- Principal component scores obtained after dimensionality reduction

Configuration :

- Number of clusters: $k = 5$ (as determined via the Elbow Method; *see Section 5.6*)
- Initialization: `k-means++` seeding
- Distance Metric: Euclidean distance
- Convergence Tolerance: $1e^{-4}$
- Maximum Iterations: 300

Evaluation :

- Intra-cluster similarity: 4.866
- Inter-cluster separation: 17.716
- Visualization: PCA plot with clear cluster boundaries (Fig. 5.6)

The clusters obtained in this stage served as the basis for the decision heuristics in Table 5.2, which specify whether particular obstacle types should be avoided or traversed.

5.8.2 Regression Model: Multiple Linear Regression (MLR)

To quantify the severity of a potential collision, an MLR model was trained on synthetically generated collision records. The model estimates the expected damage score as a continuous value.

Input Features :

- Robot Velocity (m/s)
- Impact Angle (rad)
- Impact Force (kN)
- Post-Impact Vibration (g)
- Obstacle Type
- ...
- ..

Target Variable :

- Predicted Damage Score

Training Setup :

- Dataset Size: 300 samples
- Train-Test Split: 80/20
- Tool: MATLAB `fitlm()` function, ordinary least squares.
- Loss Function: Mean Squared Error (MSE)
- Metrics: RMSE, R^2

Results :

- RMSE: 0.52 (Test records 20%)
- R^2: 0.91
- Residuals: approximately normal
- Multicollinearity: negligible

The trained regression model was coupled with the PSO-based planner to dynamically evaluate the safety of traversing obstacles. Fig. 5.7 illustrates the accuracy of the model on unseen test data.

5.8.3 Formulation of Clustering

Clustering is a key technique in unsupervised learning, used to group data points into sets based on similarity. In this work, obstacles were clustered according to features derived from both physical properties and sensor measurements, such as size and hardness.

Formally, K-Means seeks to partition n observations $\{x_1, x_2, ..., x_n\}$ into k clusters by minimizing the within-cluster squared distances:

$$\underset{S}{\operatorname{argmin}} \sum_{j=1}^{k} \sum_{x_i \in S_j} \|x_i - \mu_j\|^2 \tag{5.10}$$

where:

- $S = \{S_1, S_2, ..., S_k\}$ are the cluster sets,
- $\mu_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i$ is the centroid of cluster S_j,
- $\|x_i - \mu_j\|$ is the Euclidean distance between a point and its centroid [3].

The resulting groups represent obstacle categories with distinct risk profiles. These were later used in the decision-making stage of path planning. Please refer to Algorithm C.16 for the complete pseudocode in Appendix C.

Table 5.2: Decision heuristics after clustering

Type	**Decision Rule (damage d)**	**Action**
Hole	$d <$	Traverse
Peak	$d >$	Avoid
Liquid	$d++$	Traverse
Other Robot	–	Avoid (safety rule)
Structures	$< d <$	Traverse or Avoid

5.8.4 Formulation of Regression

MLR expresses the relationship between a continuous target variable and several predictors as a linear combination of input features [105]. The general formulation is:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon \tag{5.11}$$

where:

- Y: dependent variable (predicted damage),

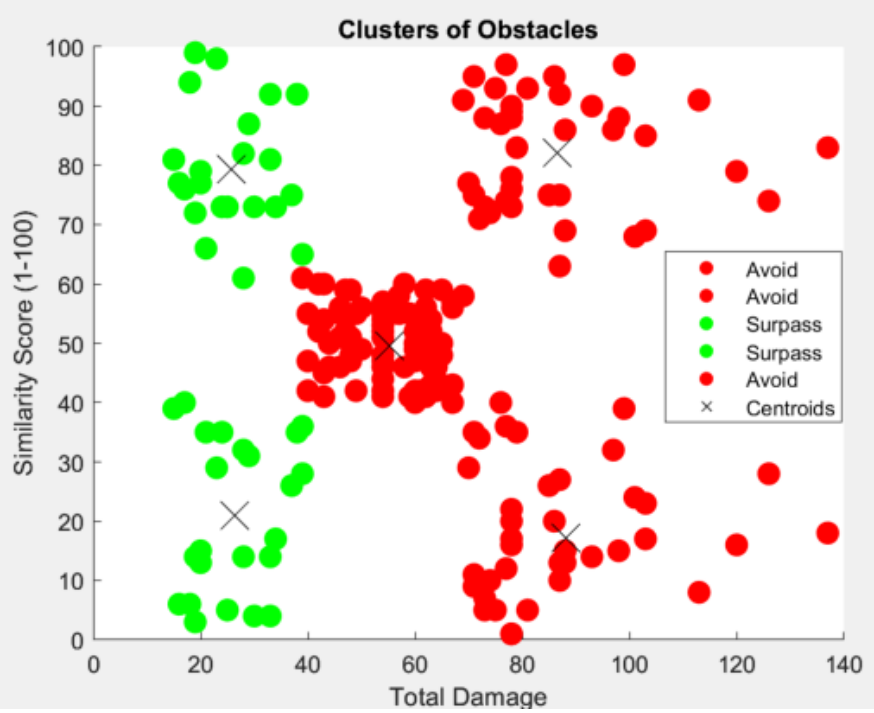


Figure 5.6: Clustering of obstacles using PCA reduced features.

- $X_1, X_2, \ldots, X_n$: independent predictors (obstacle features),
- β_0: intercept,
- β_i: regression coefficients,
- ε: error term.

The MLR assumptions (*linearity, independence, homoscedasticity, normality of residuals, and low multicollinearity*) were verified to ensure robustness [100]. Please refer to Algorithm C.15 for the complete pseudocode in Appendix C.

In this thesis, features such as obstacle dimensions, severity, and type encoding were used to predict expected damage. The evaluation showed a high predictive accuracy, with R^2 exceeding 0.90.

The clustering module provides categorical grouping of obstacles for high-level decision rules, while the regression model produces fine-grained damage estimates. Integrated with the PSO-based planner, these models allow robots to:

- generalize obstacle categories into traverse/avoid strategies,
- quantify risk in dynamic for each encountered obstacle,
- adaptively choose paths that balance efficiency and safety.

This combined architecture forms the learning-driven backbone of the proposed navigation system.

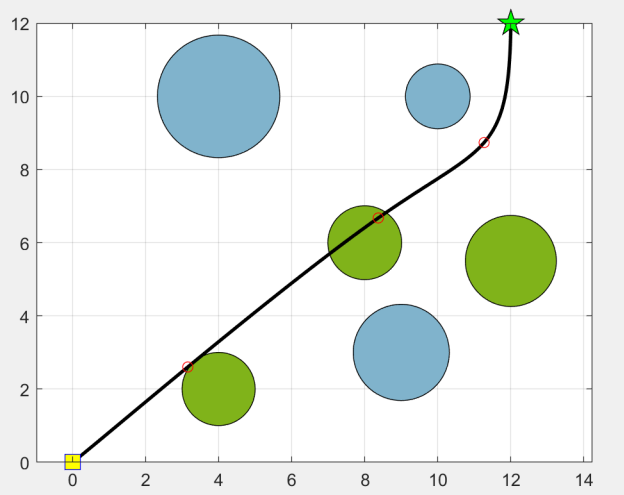


Figure 5.7: Regression model predictions.

5.9 Summary

This chapter introduces the learning-driven core of the proposed navigation framework. Starting from raw collision data, dimensionality reduction via PCA enabled a more efficient representation of obstacle features, preparing the groundwork for machine learning. On this foundation, K-Means clustering was used to group obstacles into categories of similar structural and risk-related profiles, while MLR provided a quantitative estimate of potential damage. These two models complement one another: clustering offers high-level decision heuristics, and regression supplies precise, real-time assessments of collision risk.

The integration of these components with the PSO-based path planner yields a decision-making algorithm that no longer relies on rigid avoidance rules. Instead, the robot learns to evaluate when traversing an obstacle is necessary and when avoiding it is more efficient. This ability to balance mission efficiency with structural safety is a key departure from traditional navigation methods, where all obstacles are treated uniformly. Learning from previous collision experiences, the robot adapts dynamically to new environments, achieving a robust compromise between survivability and performance.

The significance of this approach lies in its adaptability. Experimental validation in simulated environments showed that the robot could handle a wide range of obstacle scenarios by making informed and context-sensitive decisions. The system, therefore, contributes to safer, more energy-efficient, and more mission-oriented robotic navigation.

In conclusion, the methodology presented here establishes a basis on data for autonomous decision making. It provides both the theoretical basis and the practical mechanisms necessary for robots to learn from experience, generalize across environments, and optimize navigation in real time.

The next chapter extends this framework by introducing the Emergency Framework, which further enhances resilience under dynamic and uncertain conditions.

Chapter 6

Emergency Framework in Robotic Environments

Up to this point, the first research objective of this thesis has been addressed. This chapter shifts the focus to the second objective, namely the formulation of an emergency planning framework for robotic environments. As highlighted in the Introduction, emergency scenarios in robotics have received considerably less systematic attention compared to navigation and path-planning problems. To bridge this gap, we cast emergency response as a constrained multi-objective problem and align the framework with established incident management and robot safety standards [80, 77, 78].

Robots have been adopted in the logistics, manufacturing, and service industries to carry out tasks such as autonomous delivery and real-time inspection [207]. With sophisticated sensors, enhanced algorithms, and artificial intelligence capabilities, these robots efficiently and precisely manage tasks such as cleaning, maintenance, inspection, and delivery [184, 164]. The integration of robots into operational settings brings considerable advantages, such as savings in labor costs, increased safety, and enhanced productivity. In the agricultural use case, proactive robot relocation based on early warning signs has proven to be crucial. This experience confirmed the need for scenario-based planning rather than relying solely on general emergency templates [23].

The realm of robotics faces a range of unfavorable circumstances, some of which result in minor impacts, while others lead to severe consequences that can cause substantial damage or render the surroundings inoperable. Hence, it is crucial to establish a contingency strategy to be prepared for unforeseen incidents that could disrupt activities or endanger the well-being of staff and resources. Each robotic setting has different characteristics and demands due to variations in operational domains and types of robots, highlighting the need for customized emergency protocols [143].

Developing an emergency strategy from the ground up presents significant obstacles and susceptibilities. One key difficulty is ensuring the safe retrieval of all robots in a responsible manner. The loss or damage of robots during tasks can complicate the extraction procedure, leading to intricacies and delays [13]. Furthermore, involving various stakeholders, such as different nations or entities, requires transparent communication, shared goals, and a clearly defined emergency strategy to facilitate seamless and structured operations [37]. In addition, safeguarding the confidentiality of the data and equipment used by stakeholders

is paramount, given the value and sensitivity of this information [143]. Proper handling of hazardous materials or waste produced during activities is also crucial. Addressing these hurdles requires thorough planning, synchronization, and implementation to guarantee a secure, streamlined, and efficient process.

When faced with such challenges, it is essential to establish a guide to construct a resilient and flexible emergency strategy. *However, formulating a universal solution for robotic domains is impractical because of the different scenarios that could emerge in various contexts*. Hence, this research suggests a top-level structure that acts as a guide for future emergency strategies across all robotic sectors. This structure offers a broad conceptual overview and fundamental procedures for a suitable strategy. Its efficacy lies in its integration with the precise requirements of the robotic setting. Fig. 6.1 shows how the suggested framework integrates seamlessly into a comprehensive strategy.

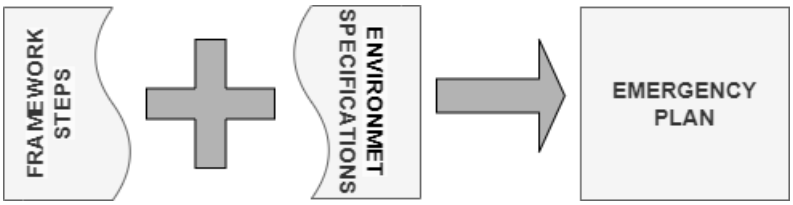


Figure 6.1: A Figure of transformation of the proposed framework to an emergency plan.

6.1 Proposed Framework

Creating an efficient emergency strategy for the field of robotics presents a challenge due to the diverse nature of the robotic sectors and the variety of robot types involved [86, 163]. In order to tackle this obstacle, a structured approach has been suggested to offer companies a practical guide for formulating their own emergency strategies. This model acts as a comprehensive guide, aiming to encompass all essential elements and arm organizations with the necessary resources to deal with emergencies in robotic settings. It recognizes the distinct features and hazards associated with various robotic fields, ensuring that the emergency strategy satisfies these specific requirements. By adhering to this guideline, companies can increase their readiness and response capabilities, empowering them to efficiently manage emergencies within the realm of robotics.

The suggested framework presents a promising strategy for emergency planning within the realm of robotics. Its value lies in its ability to seamlessly integrate with the requirements of the robotic setting. Only by customizing the framework to align with the unique attributes of the robotic domain can a truly effective emergency plan be devised. Taking into account the distinct features of the robotic sector, such as operational protocols and possible hazards, organizations can adjust their emergency response strategies accordingly. This advancement turns the framework into a useful tool that aligns the emergency plan with the specific demands and obstacles of the robotic field, thus improving readiness and efficiently reducing potential risks.

Before we dive into the framework, we first need to recognize the different types of emergencies that can impact robotic systems. This is important because knowing what could go wrong is the first step in building strong practical solutions [7]. Every crisis comes with its own challenges, and by understanding them, we can craft emergency strategies that are both resilient and thorough. *Our framework takes this into account,*

covering all critical risks in robotics, from technical failures to external threats. With this proactive mindset, organizations can create flexible emergency plans ready to handle anything that comes their way.

Emergency Planning Problem Let $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$ be the set of possible emergency scenarios in the robot operating environment $\mathcal{C} \subset \mathbb{R}^n$. Each emergency $e_i \in \mathcal{E}$ is described by:

$$e_i = (s_i, r_i, p_i) \tag{6.1}$$

in equation 6.1:

- s_i: complexity level,
- r_i: maximum allowed response time,
- p_i: physical or logical components affected by the event.

Let $\mathcal{F} = \{f_1, f_2, \ldots, f_\ell\}$ be the set of available emergency response actions for the robotic system. The objective is to select the optimal response:

$$f^* = \arg\min_{f_j \in \mathcal{F}} \left[\alpha \cdot \frac{T(f_j, e_i)}{T_{\max}} + \beta \cdot \frac{R(f_j, e_i)}{R_{\max}} + \gamma \cdot \frac{C(f_j, e_i)}{C_{\max}} \right] \tag{6.2}$$

The optimization is subject to the time constraint:

$$T(f_j, e_i) \leq r_i, \quad \forall f_j \in \mathcal{F},\ \forall e_i \in \mathcal{E} \tag{6.3}$$

where:

- $T(f_j, e_i)$: estimated execution time of response f_j under emergency e_i,
- $R(f_j, e_i)$: residual risk remaining after applying f_j,
- $C(f_j, e_i)$: cost of applying f_j (like energy, resource usage),
- α, β, γ: weighting coefficients determined by mission priorities.

All terms are normalized to a common scale $[0, 1]$ using maximum plausible values $T_{\max}$, $R_{\max}$, and $C_{\max}$ to ensure unit consistency.

Standards Alignment

This formulation represents emergency response selection as a multi-objective optimization problem and is largely adapted from [80, 81], balancing severity, urgency, and impact against available system capabilities. Domain-specific constraints instantiate these requirements at implementation time.

Emergencies in Robotic Fields In robotics, an *emergency* is an unexpected event requiring prompt action to protect humans, robots, or the environment. Based on standard taxonomies and recent surveys, critical categories include:

1. **Robot Malfunction**: A robot can experience a mechanical or electrical failure that jeopardizes its operation or poses a risk to nearby people. This could include issues such as loss of control, software failures, sensor failures, or power supply problems.

2. **Human-Robot Interaction Accidents**: Robots working in close proximity to humans can accidentally cause harm or injury. For example, a robot arm could collide with a person, leading to physical damage.

3. **Environmental Hazards**: Robots deployed in hazardous environments, such as nuclear power plants, chemical facilities, or disaster zones, can face emergencies related to leaks, spills, explosions, or other dangerous conditions. Robots may be tasked with mitigating emergencies or assisting in rescue and recovery operations.

4. **Unforeseen Obstacles or Events**:Robots operating autonomously or in complex environments may encounter unexpected situations that require immediate response. These could include sudden changes in terrain, objects that block the robot's path, or unanticipated events that demand adaptive decision-making. [171, 246, 96]

In each of these instances, it is essential to have the necessary emergency procedures and safety measures in place to efficiently deal with and control the circumstances [171]. These measures could encompass emergency shutdown protocols, failsafe systems, human oversight, remote control functionalities, and specialized teams trained to handle robotic crises.

6.2 Pillars of the Framework

The suggested framework aims to define a predetermined series of steps and measures to be executed in the event of an emergency [171]. *It comprises five primary components that need to be customized according to the particular types of robots and the robotic environment.*

Risk Assessment and Decision-Making : We adopt [81, 79] risk workflow; hazard identification, risk analysis, risk evaluation, and risk treatment traceable controls. Decision-making is driven by the quantified risk picture and mission objectives; alternatives are compared based on safety, operational impact, and resource cost, with explicit acceptance criteria and escalation paths.

When conducting robot risk assessments in various settings, organizations must consider the unique risks associated with each environment. In industrial settings, these risks can include incidents such as robot-human or robot-equipment collisions, mechanical hazards such as entrapment or crushing, electrical hazards such as shocks or fires, risks related to material handling, and environmental factors such as extreme temperatures or exposure to hazardous substances [38].

The healthcare sector must carefully assess risks, primarily concerning patient safety, such as potential diagnostic inaccuracies and infection prevention protocols. Moreover, safeguarding privacy and data integrity is crucial, especially when robots are involved in managing security or safety information, which could lead to malfunctions or compromise safety. Furthermore, the dynamics of human-robot interaction pose a risk to effective communication and comprehension between robots and healthcare professionals [22].

Public areas present unique risks for robots, such as dealing with issues such as crowd control problems such as unintentional collisions or actions that may cause panic [56]. Introducing robots with cameras or sensors into these spaces raises concerns about privacy and security, necessitating evaluations to avoid unauthorized surveillance or data leaks. During emergencies, it is essential to assess the risks that robots face in unpredictable surroundings, dangerous situations, or when there is minimal human oversight.

Working in dangerous settings brings about additional hazards, such as robots being exposed to chemicals in toxic surroundings or facing radiation in ionized radiation environments [102]. Severe circumstances, such as extreme temperatures, high pressure zones, or underwater tasks, require risk evaluations to guarantee the safe operation of robots under those conditions.

Risk Assessment

- *Identify Potential Risks: [79]*: enumerate mechanical, electrical, HRI, environmental, and cyber-physical threats. Define triggers with green/yellow/red thresholds.
- *Establish an Evacuation Plan*:Evacuation plans that consider specific key components for service robots. Develop a plan of evacuation, ensure that all members of the team are aware of it, evaluating key parts of your service robots. Ensure that this is done correctly.

Communication Protocols and Channels : Efficient communication procedures and channels play a crucial role in emergency strategies involving robots [216]. It is vital for personnel and stakeholders to have access to clear and prompt communication to stay informed about the situation and respond effectively during emergencies. This requires a clearly defined communication protocol that delineates the duties of emergency response personnel and specifies the communication channels to be used. *In emergency situations, having a clearly structured communication plan ensures faster response and minimizes confusion.*

The communication protocol must describe the people responsible for initiating the communications, the recipients to be notified, and the methods to distribute the information [211]. It is crucial to incorporate alternative communication channels in the event that the primary ones are compromised or inaccessible. Regularly testing and revising communication protocols is essential to ensure their effectiveness and suitability for evolving circumstances.

In case primary communication channels become compromised or inaccessible, organizations have the option to set up alternative communication channels to ensure connectivity and information flow [183]. These alternatives may consist of duplicate networks from various providers, satellite communication setups, two-way radios, public platforms such as social networks, phone trees, and mesh networks [131].

In order to guarantee the effectiveness and flexibility of these communication protocols, it is essential that organizations adhere to a series of procedures [239]. Routine testing through drills and simulations serves to identify vulnerabilities, verify correct operation, and familiarize staff with alternative communication techniques. Scenario-based exercises replicate disturbances or crises to assess the strength of protocols and pinpoint areas that need improvement. Educational sessions and awareness initiatives inform employees about emergency communication channels and protocols, improving their readiness and comfort in utilizing alternative channels [171].

Periodic assessments of duplication, dependability, capability, and suitability of backup pathways are crucial to ensuring their effectiveness and alignment with advancing technologies and organizational requirements. Subsequent assessments following incidents allow organizations to pinpoint problems, assess the performance of backup pathways, and make the necessary improvements using the insights gained. By setting up precise communication protocols and channels, organizations can guarantee their capacity to react promptly and effectively to emergency situations, thereby reducing the impact on operations and personnel.

Communication Channel

- *Ensure that Robots are Ready*: Make sure that the robots are easily accessible and do not be blocked by obstacles. In addition, they have backup power sources.

Resource Allocation and Coordination : Effective resource allocation and coordination play a vital role in the development of emergency strategies for robots. A thorough understanding of the existing resources and their optimal distribution is crucial to solving this problem. This involves evaluating the personnel, equipment, and materials needed to execute the emergency plan [215].

The strategy must clearly define the duties and obligations of the staff involved in the emergency response and establish protocols for managing the distribution of resources. This could include setting up a command center or control room to supervise the response and guarantee effective resource management. *Partnerships and agreements with outside entities can also be formed to acquire additional resources or assistance in times of emergency.*

Furthermore, the strategy should incorporate protocols to monitor and assess the efficiency of resource distribution and organization to ensure optimal use. Continuous monitoring and evaluation allow for necessary adjustments and enhancements to be implemented [99].

Through the implementation of efficient resource allocation and coordination protocols, organizations can guarantee their ability to respond quickly and effectively to emergency situations, thus reducing the impact on both operations and personnel.

Resource Coordination

- *Train Team Members*: Ensure that all team members are trained in the behavior of emergency situations. Assign specific roles to your team members to the evacuation robots. For example, someone might be responsible for shutting down or disconnecting robots from power sources.
- *Test the Plan*: Even if the plan is created very well, it should be tested in advance and prospective obstacles removed.

Implementing the Emergency Plan : Implementing the suggested framework for emergency strategies can present difficulties for institutions. These difficulties include the distribution of resources, compatibility with technology, training and adaptation to new environmental conditions, regulatory and legal aspects, adaptability, challenges in collaborating with stakeholders, and routine evaluation and review [52].

To address these difficulties, organizations can pay attention to emergency readiness in financial planning, explore sources of funding, and engage in partnerships with external parties. Rigorous compatibility assessments, cooperation with technology specialists, and frequent communication with service providers can help tackle technical challenges through thorough compatibility evaluations. In addition, organizations should offer comprehensive training, perform routine exercises, periodically evaluate the plan, and develop user-friendly manuals to improve employees' understanding of backup systems [86].

The implementation stage of the emergency plan in the field of robotics requires the execution of pre-established procedures when a crisis arises. Rapid decision making, efficient coordination, and precise communication are crucial to effectively execute the plan. During this phase, the emergency response plan is activated and predefined procedures are executed to protect personnel and robotic assets [171].

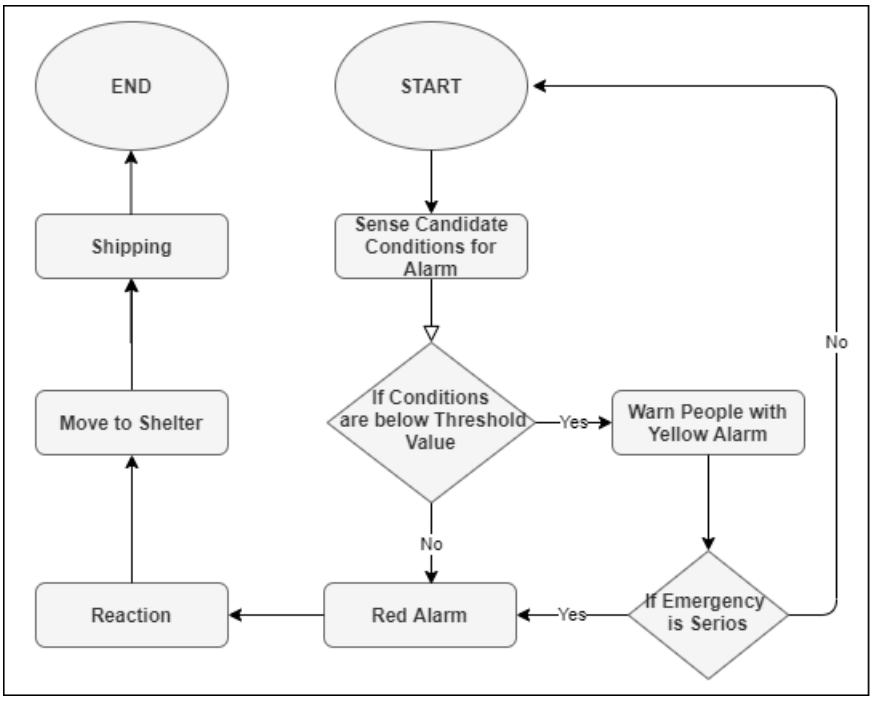


Figure 6.2: A flow of implementing the emergency plan.

- *Sense Environment*: Robotic fields must be continuously monitored. Sensors should collect environmental information and provide it to a decision system.
- *Decide to Alarm*: Various levels of symptoms can be collected from the environment. Therefore, there must be a threshold for this type of measurement. For example, temperatures can increase abnormally but still be normal on summer days. Therefore, the defined threshold helps determine whether it is an emergency or not. When the threshold is reached, the system decides that the emergency is on; otherwise, it warns the robotic field staff to decide. The first is the red alarm and the second is the yellow alarm, respectively.
- *Initiate the Emergency Plan*: After the emergency is declared and the safety of the team is ensured, the plan should be applied as soon as possible.
- *Evacuate the Robots*:Evacuate robots from the assembly area as quickly and safely as possible. Then, if necessary, transport them to secure locations.

Emergency Plan

By adhering to these procedures, companies can aim to achieve the successful implementation of the emergency strategy, guaranteeing that all parties involved are ready and able to respond promptly and appropriately to emergencies in the field of robotics. Fig. 6.2 shows the framework process proposed in this chapter.

Post-Emergency Analysis and Feedback : Conducting a thorough evaluation is essential to identify areas for improvement and refine response strategies after the situation is under control and hazards, such as robots, have been cleared from the site [86]. This assessment should examine key aspects such as response protocols, communication effectiveness, resource allocation, and teamwork.

Providing clear and constructive feedback to responders, stakeholders, and partners is equally important. Feedback should acknowledge what worked well, point out strengths, and suggest actionable improvements. These insights must then be used to update emergency plans, ensuring that the lessons learned are put into practice. By consistently evaluating responses and sharing meaningful feedback, organizations can strengthen their emergency preparedness and adapt more effectively to future crises.

- *Review Robot Status*: Review the status of robots after an emergency to ensure that they are working properly and do not have damage. You can do this.
- *Conduct an Evaluation*: Perform an emergency evacuation evaluation to determine what worked well and what could be improved. Then update your plan.
- *Re-Test the Plan*: Regularly check the evacuation plan to ensure that it is still effective and that all team members are aware of it.
- *Provide Additional Training*: As needed, provide team members with additional training to ensure that they are ready for future emergencies.

Steps of Framework :

The framework prescribes: (i) risk/vulnerability identification, (ii) response protocols with clear triggers and roles, (iii) redundant communication, (iv) training and drills, and (v) monitoring and continuous improvement. Security, asset handling, and waste management are integrated as constraints. Fig. 6.3 summarizes the phases **before**, **during**, and **after** an incident.

Advantages of the Framework : The proposed framework brings several practical advantages to organizations that work with robotic systems. One of its key strengths is **flexibility**; it can be adapted to different types of robots and operational environments. This helps organizations avoid designing separate emergency plans for every situation, saving both time and effort. Instead, they can follow a unified approach while still making adjustments based on the specific risks in their domain.

Another important benefit is that the framework can support organizations that **lack prior experience** in emergency planning. Especially for smaller teams or new deployments, having a structured guideline to start from can reduce the burden of building a strategy from scratch. It offers a clear sequence of steps that makes it easier to identify potential risks, assign responsibilities, and take timely action in critical situations.

The framework is also designed to **evolve**. After an emergency occurs, organizations can revise their response procedures based on real-world feedback. This ongoing learning process ensures that the emergency strategy remains relevant and continues to improve over time. Using this framework, teams can better prepare for unexpected events and strengthen both safety and coordination throughout their robotic operations.

6.3 Implementation and Case Studies

Strategies and Considerations When implementing the framework, there are several considerations that must be taken into account [13]. Organizations should:

1. Ensure that the framework is tailored to the specific needs and characteristics of the robotic field in question. For example, the framework may need to be modified for different types of service robots or emergency scenarios.

2. Ensure that all relevant parties are included in the implementation process. Working with manufacturers, first responders, and end users may be necessary to create and test the framework. To ensure that all stakeholders can successfully apply the framework, it can also be important to offer training and education on it.

3. Consider the technical requirements of implementing the framework, such as the need for specialized communication tools or equipment. This may require additional investment or upgrades to the infrastructure to ensure that the framework can be implemented effectively in emergency situations.

4. Ensure the ongoing evaluation and refinement of the framework: It remains relevant and effective over time. Regular reviews and updates can help identify areas for improvement and ensure that the framework can keep up with changes in technology, emergency scenarios, and stakeholder needs.

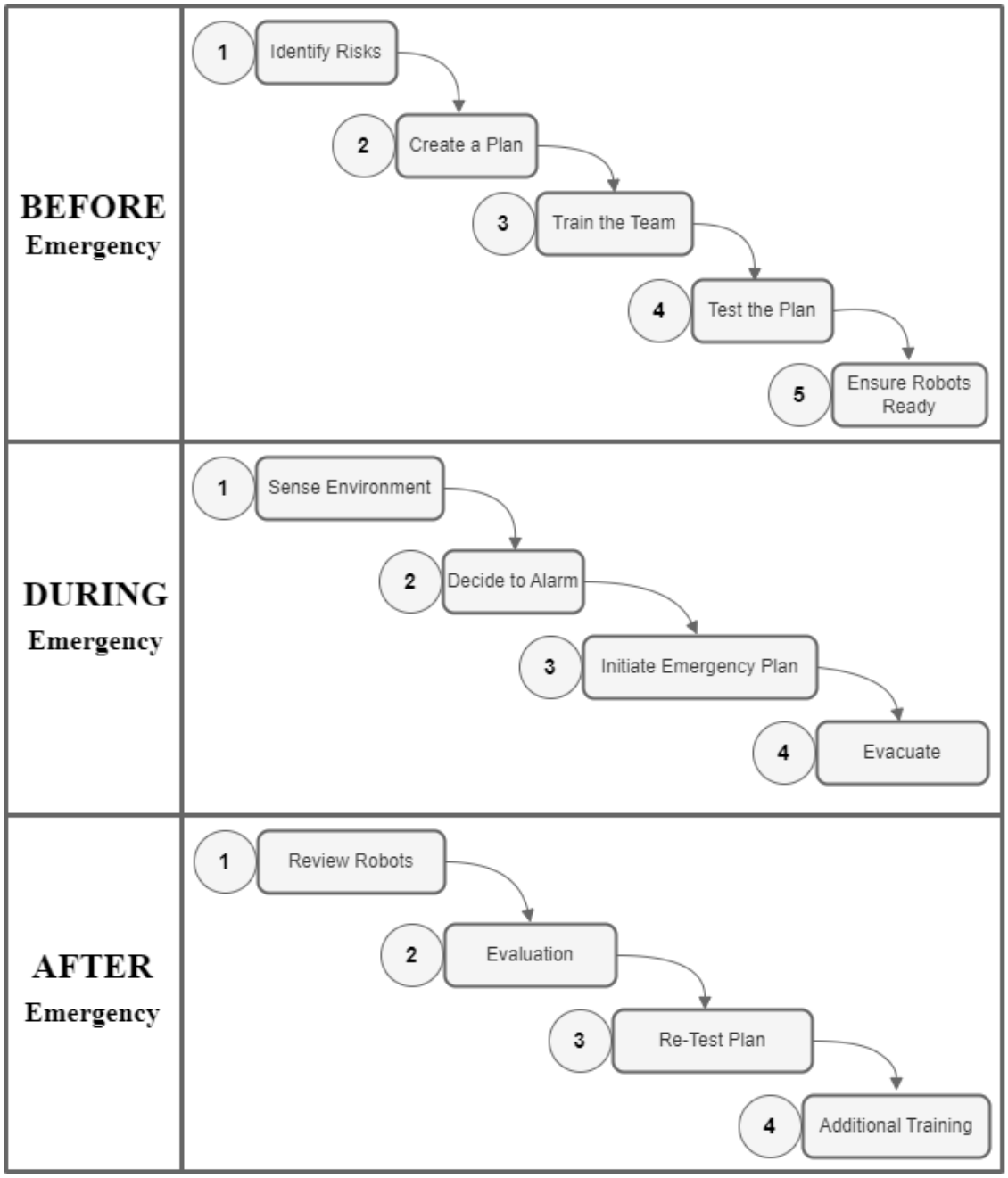


Figure 6.3: The story of an emergency plan

Emergency in Agricultural Fields Agricultural regions are inherently vulnerable to sudden and unpredictable events [227], a factor that has directly shaped the design of our proposed framework. To illustrate this, Table 6.1 summarizes representative examples of agricultural emergencies. Consider, for example, a farm that relies on autonomous robots for crop management. When a severe thunderstorm is forecast, bringing strong winds and heavy rainfall, the risk of damage to robots and nearby infrastructure, as well as potential harm to individuals, becomes significant. In response, farmers proactively implement the framework by relocating the robots to a designated safe area, thus mitigating the anticipated risks.

Table 6.1: Representative agricultural emergencies [149].

Natural		**Human-Based**		
Floods incidents, Hurricanes, Wildfires, Lightning strikes,	Severe dust storms, Severe winter, Earthquakes, Tornadoes	Wildfires explosions or fires, Amputations, Power failures	Animal Handling incidents, Rotating/moving incidents, Equipment Vehicle Accidents	Workplace violence, Accidental poisoning, Chemical releases or spills

Implementing An Emergency Case: Here are the step-by-step emergency case implementations.

Risk assessment and decision-making: Farmers assess the severity of the storm and determine that it poses a significant danger to both the robots and the farm. To protect them from the storm, they opted to relocate the robots to a nearby storage facility.

Communication protocols and channels: Farmers use radio communication to discuss the evacuation plan with their peers. They also inform local emergency services in case of need.

Resource allocation and coordination: To secure the robots, farmers load them onto a truck and transport them to the storage facility, designating workers to assist with the evacuation effort. In addition, they cooperate with the facility to ensure that there is sufficient space and equipment for the robots.

Implementing the Emergency Plan: It is important to continuously monitor the agricultural environment. Depending on the structure of the environment, this could be achieved in various ways. In particular, farmers or specialized sensors can detect thunderstorms manually or automatically. If the severity of the symptoms is extreme, a red alarm may sound; otherwise, a human should be warned before taking any further action. The strategy should commence if it is determined that the situation qualifies as an emergency under the yellow alarm. Robots and assets should be transported to the assembly area before being shipped to a secure location, if necessary.

Post-emergency analysis and feedback: Farmers carry out a post-emergency review to evaluate the success of the evacuation effort after the storm has passed. They provide feedback to their team and other stakeholders and point out areas that need improvement, such as the requirement for stronger communica-

tion channels and more detailed evacuation plans. To be better prepared for potential emergencies, they also update their emergency plan.

Results and Evaluation of the Implementation The results of implementing the proposed framework will naturally vary depending on the application domain, the types of emergencies encountered, and how the plan is executed under real conditions. *In field tests such as the agricultural case study, the ability to relocate robots in time and prevent physical damage proved to be a critical success factor.*

Rather than relying solely on predefined procedures, the **responsiveness** of the teams and their familiarity with the plan made a noticeable difference. Metrics such as evacuation time, robot preservation rate, and staff coordination were helpful in evaluating performance, but informal user feedback also revealed gaps, particularly in communication tools and situational awareness.

Another key factor was **how well the framework could be adjusted** when conditions changed. In practice, emergency scenarios rarely unfold exactly as expected, so flexibility and regular plan updates were essential. Teams that had conducted drills or had scenario-specific tweaks to the framework performed significantly better.

Beyond immediate response, reducing **operational downtime** and protecting **staff safety** were significant benefits. In the agriculture use case, early detection and action helped maintain continuity with minimal disruption. These observations support the idea that the framework, when applied thoughtfully, can offer both short-term and long-term value.

Ultimately, implementing such a framework is not a one-time task but a process. It requires consistent evaluation, revisions based on experience, and commitment from all stakeholders. With this mindset, organizations can gradually improve their emergency readiness as both technology and operational environments evolve.

6.4 Summary

This chapter introduces a structured framework for emergency planning in robotic environments, conceptualized as a constrained multi-objective optimization problem. By systematically combining risk assessment, communication protocols, resource coordination, implementation guidelines, and post-emergency evaluation, the framework provides both a theoretical foundation and practical guidance for organizations operating in diverse robotic domains.

The agricultural case study illustrated how the framework can be applied in practice, showing that proactive planning and flexible adaptation can significantly reduce damage, minimize downtime, and enhance overall resilience. A key strength of the framework lies in its adaptability: it is not a fixed template but rather a scalable methodology that can be tailored to sector-specific risks and continuously improved through feedback and real-world experience.

In conclusion, this framework establishes a solid foundation for emergency preparedness in robotics, supporting safer, more coordinated, and more efficient operations. It also lays the groundwork for future

research on domain-specific adaptations and integration with data-driven navigation models presented in earlier chapters.

Having introduced the proposed methodology in the previous chapter, the next chapter puts it to the test through a series of simulation-based experiments. These experiments are designed to evaluate the performance, adaptability, and robustness of the system under diverse scenarios and environments.

Chapter 7

Experimental Results and Analysis

This chapter evaluates the proposed method under controlled and progressively complex simulations. We report design, metrics, results, and statistical tests with an emphasis on reproducibility. Claims are limited to the tested conditions; we discuss limitations and failure modes where observed.

The structure of the chapter is as follows: we begin with a detailed description of the **Experimental Setup**, followed by a discussion of the **Evaluation Metrics**. The results are then presented at multiple test levels and optimization techniques and further validated using **ANOVA Statistical Analysis**. A realistic application scenario is then introduced to highlight the practical value, before concluding with a broader **Discussion** and a concise **Summary**.

7.1 Experimental Setup

The experimental setup for our simulations, conducted using MATLAB, was meticulously designed to investigate the behavior and performance of robotic systems in dynamic environments with varying obstacle configurations and robot densities. This setup consisted of several key components, including the simulation environment, the robotic agents, and the various obstacle layouts.

Simulation Environment: The simulation environment was implemented using the MATLAB toolboxes, which provide a versatile platform to model complex robotic systems. The environment consisted of a two-dimensional workspace with customizable sizes, allowing us to simulate scenarios of different scales and complexities. In addition, we integrated structure-based modeling to accurately replicate interactions between robots and obstacles, ensuring realistic simulation outcomes.

Robotic Agents: The robotic agents, representing autonomous entities within the simulated environment, were modeled using MATLAB's robotic toolbox, *specifically in third-level tests*. Each obstacle was shown with different colors to clarify decision-making in *first and second level tests*.

Obstacle Layouts: We explored a diverse range of obstacle layouts, varying in size, shape, and density, to assess their impact on the performance and efficiency of robotic systems. These obstacles were strategically placed within the workspace, simulating real-world scenarios encountered in environments such as warehouses, factories, urban landscapes, and narrow robotic paths. By manipulating the number, size, and

distribution of obstacles, our objective was to examine how different configurations influence robot navigation, collision avoidance, and task completion.

Experimental Scenarios: To comprehensively evaluate the behavior of robotic systems under various conditions, we conducted simulations in multiple experimental scenarios. These scenarios encompassed different combinations of obstacle layouts, ranging from sparse to dense configurations, as well as varying numbers of robotic agents.

Simulation Capabilities

The simulation tool was designed to support:

- Importing pre-defined map data from .xlsx files,
- Real-time path planning using selected algorithms,
- Collision simulation with dynamic/static obstacles,
- Logging per-run CSV (path, time, length, total damage, avoids).
- Export figures (.png), raw logs (.xlsx).

Through this well-designed experimental setup, we sought to gain a deeper insight into the challenges and opportunities inherent in the deployment of robotic systems in dynamic and cluttered environments. By systematically exploring the interplay between obstacle layouts, robot densities, and environmental conditions, our objective was to inform the design of more robust, adaptive, and efficient robotic systems capable of operating effectively in real-world settings.

7.1.1 Application Description

We implemented and tested our proposed decision-making framework within the *AnticipatorySim* MATLAB simulation platform [183, 174], which provides a comprehensive environment for robotic navigation experiments. While the simulation infrastructure was adopted from existing work, our research introduces significant algorithmic contributions:

- **Obstacle Modeling Module**: Supports insertion of static, dynamic, and traversable obstacles with adjustable parameters such as shape, size, position, and damage potential. Obstacles can be defined manually or loaded from external files.
- **Path Planning Engine**: Incorporate multiple planning algorithms including A-star, PSO, RRT, and PRM. Each planner can be configured with different parameters and the resulting path is evaluated using the unified cost function (*based on length and predicted damage*).
- **Damage Estimation Model**: Apply clustering and regression techniques to infer potential damage scores for each obstacle encountered along a candidate path. This enables risk-aware path selection.

- **Simulation Visualizer**: Graphical interface to simulate the robot's movement in a 2D grid environment. Visualizations include obstacle locations, robot paths, damage, and decision points.
- **.mlapp GUI Interface**: The MATLAB App Designer interface allows for interactive control over simulation parameters, such as number of obstacles, planning algorithm, robot start and goal position, and damage limits. The results are exported in both the chart and the Excel format for further analysis.
- **Output and Logging**: The application stores all simulation results, including path coordinates, damage values, planning times, and decision metrics, in structured files to enable statistical post-analysis.

This tool was used to run third-level experimental setups in this chapter. The modular design allows for easy expansion to support new path planners, obstacle types, and learning models in future work.

Platform Features

MATLAB R2022b (Statistics/Optimization Toolboxes), Windows 10, 16 GB RAM, single thread unless noted. Each scenario uses 10 independent runs with fixed random seeds $\{1, \ldots, 10\}$. Figures show one representative run.

7.2 Evaluation Criteria

To evaluate the proposed system, we employed a set of key performance metrics [183] (see Table 7.1), enabling a standardized quantitative analysis of path planning, obstacle handling, and decision-making. All metrics were calculated per-run, and the final results represent the mean values across 10 runs. Within this framework, *damage* is quantified as a collision per-obstacle. A trial is deemed successful under two conditions: the agent must achieve the goal successfully, and its accumulated *damage* must not exceed the maximum allowed value D_{max}.

Table 7.1: Evaluation metrics summary

Metric	Description
Completion Time	Total time required to reach the goal from the start point. Lower values indicate more efficient planning.
Damage Score	Aggregated severity of all obstacles traversed. Lower values indicate better safety performance. The normalized value $D = D_{\text{raw}}/100 \in [0, 1]$.
Path Length	Total length of the path taken by the robot. Helps quantify detours or inefficiencies.
Success Rate	Percentage of successful goal completions without critical damage or failure.
Energy Consumption	Estimated energy use based on path length, time, and obstacle interactions.

Together, these metrics provide a holistic evaluation that balances classical efficiency indicators with safety and resilience measures that are unique to damage-aware path planning.

7.3 Presentation of Results

The effectiveness of the proposed framework was validated at three test levels, each designed to incrementally increase the complexity and stress the decision-making capacity of the system. Beyond qualitative visualizations, quantitative summaries were produced to ensure the reproducibility and reliability of the results. Tables summarizing average completion time, success rate, and standard deviations provide evidence of consistency across multiple runs.

Each test group was composed of several distinct tests, each varying in environmental elements and dimensions. These tests encompassed a variety of obstacles, including static barriers, dynamic objects, and uneven terrain, to thoroughly assess the adaptability of the framework. The testing environments ranged from simplistic setups with few obstacles to highly intricate and unpredictable scenarios, demanding sophisticated decision-making.

The results of every test were documented and evaluated according to essential performance metrics, such as the accuracy of damage prediction, the efficiency of path optimization, and the success rate of obstacle navigation. By examining the results across the three levels of complexity, we gleaned insights into the strengths and weaknesses of the proposed method under varying operational scenarios. *The subsequent sections present a comprehensive analysis of the findings, highlighting patterns and significant observations from the tests.*

7.3.1 First Level Tests

This level represented the baseline condition, with few obstacles and simple avoidance/traverse choices.

- Robots successfully completed navigation in nearly all cases, with an average success rate of **98.5%**.
- The completion time was low (avg **12.3 s**, SD = 0.8 s), reflecting the simplicity of the environment.

Fig. 7.1 presents the first test group, demonstrating the system's fundamental path planning behavior after classifying obstacles as traversable or non-traversable. For simplicity, obstacles are represented as circles color-coded by type: light red indicates obstacles that must be avoided, while green indicates those that can be traversed. It is important to note that an obstacle's damage severity is not a function of its size; while the obstacles vary in dimensions, their damage parameters are independently configured to simulate different risk levels.

The four subfigures illustrate different planning scenarios:

- **Top-Left**: The path is successfully navigated through three traversable (green) obstacles without a collision.
- **Top-Right**: The robot avoids the nearest obstacle (red) while the other two, located away from the shortest path, remain passable.

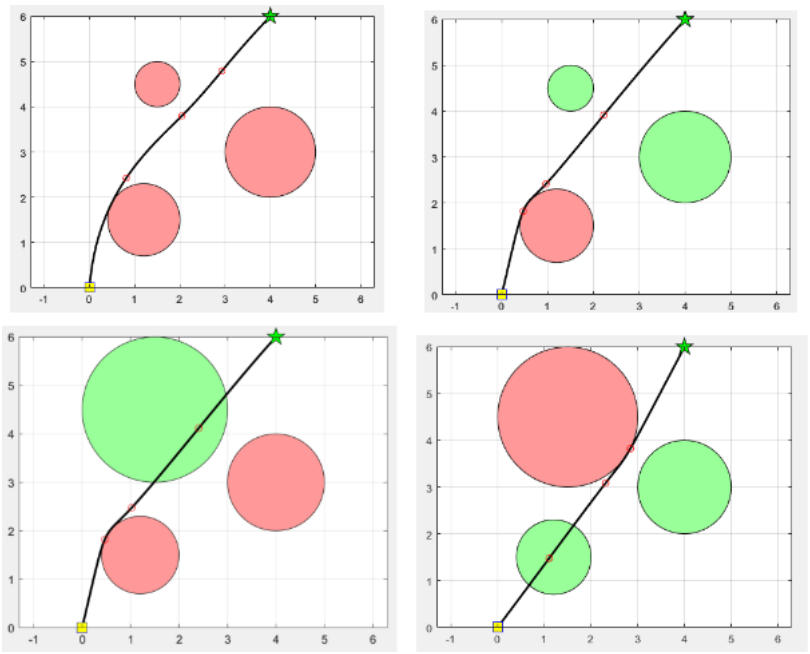


Figure 7.1: First level tests-group A.

- **Bottom-Left**: The path traverses the largest obstacle, suggesting traversal through a gaseous or aquatic region.
- **Bottom-Right**: The presence of a damaged obstacle causes a deviation from what would otherwise be the shortest route.

The methodology presented in this dissertation builds upon prior work in robotic navigation, with a specific focus on diverse and structurally complex environments. A key challenge in such settings, exemplified in Fig. 7.2, is the presence of long, narrow corridors. Navigating these can significantly increase time and energy consumption, especially when a change in path is required. This figure demonstrates the system's strategic decision-making when faced with obstacles in these constrained spaces, showing a clear preference for avoiding high-damage obstacles, even if it necessitates a detour.

The results in Fig. 7.2 can be analyzed as follows:

- **Top-Left:** The robot is presented with two routes between three walls. It correctly selects the path blocked by a low-damage (green) obstacle over the alternative containing a high-damage (red) obstacle.
- **Top-Right:** A similar configuration with altered obstacle placements further validates the consistency of the decision-making logic.
- **Bottom Row:** These panels contrast two fundamental behavior. When a corridor is blocked by vulnerable (high-damage) obstacles, the system reroutes *outside* the corridor walls. Conversely, when obstacles are traversable (low-damage), a safe path is found *within* the corridor along the walls.

These results (Table 7.2) confirm that the system generates a learning model and is able to filter obstacles accurately without introducing unnecessary avoids under low-stress conditions.

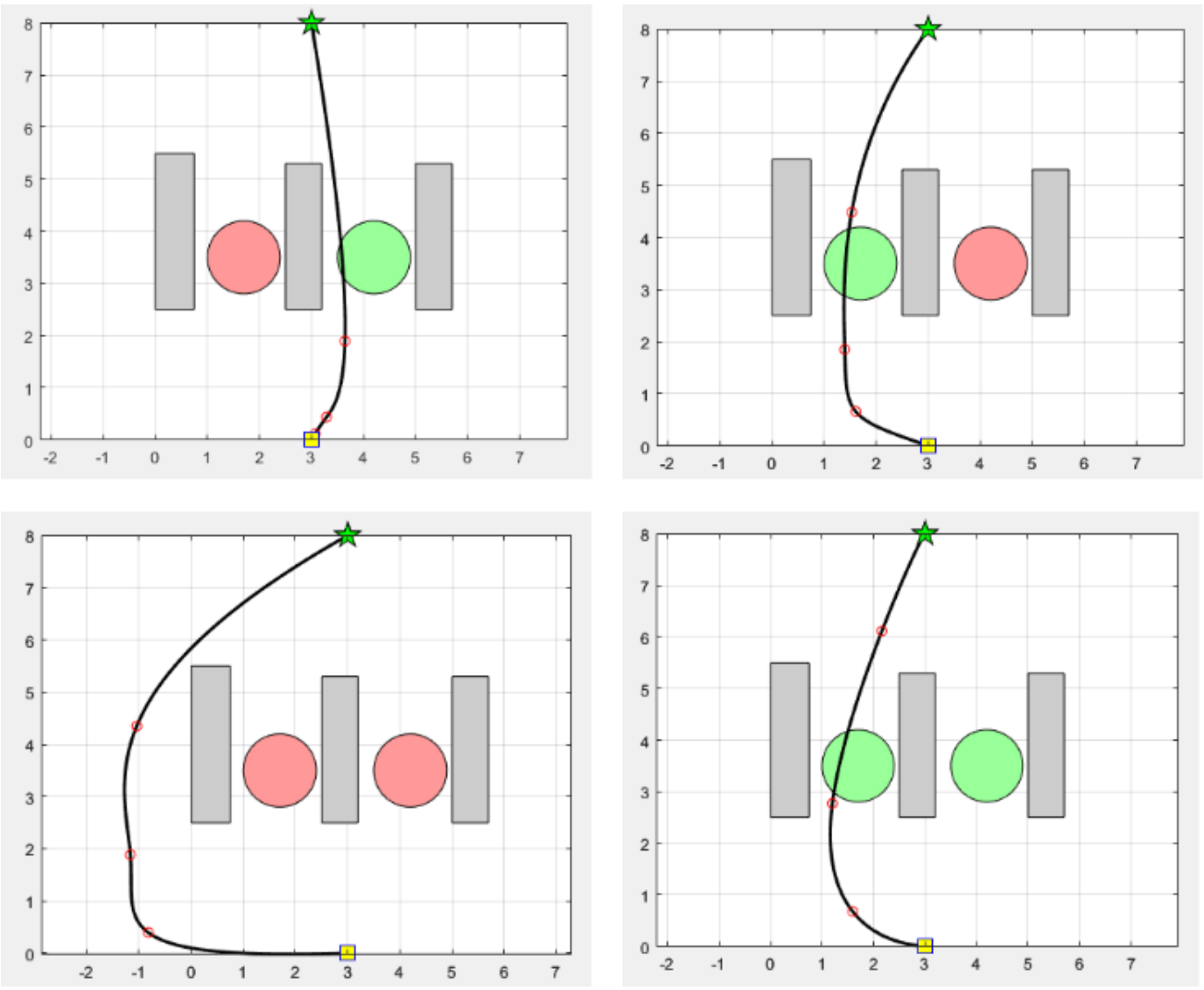


Figure 7.2: First level tests-group B.

7.3.2 Second Level Tests

At this stage, corridor-shaped structures and denser obstacles were introduced, which required robots to balance path efficiency with damage prediction.

- The success rate dropped slightly to **93.2%** (SD = 3. 4%), reflecting the increased difficulty.
- The average completion time increased to **15.6 s** (SD = 1.1 s), and the path lengths grew longer.
- The ML-based damage model consistently helped identify when controlled traversal was more efficient than detour.

In the earlier subsection, the decision-making mechanism was systematically tested. These tests used PSO to identify the path and determine the obstacles. Now, various optimization algorithms will be employed to evaluate the system, highlighting its flexibility and compatibility with different optimization techniques.

Table 7.2: Summary of first-level test results (*10 runs per scenario with PSO*)

Metric	**Mean**	**Std. Dev.**	**Success Rate**
Completion Time (s)	12.3	0.8	98.5%
Damage Score	0.12	0.03	–
Path Length (m)	11.4	0.6	–

Genetic Algorithm: The test results (Fig. 7.3) of the GAs optimization test illustrate the effectiveness of the strategy, showing that the generated solutions maneuver through the environment effectively by overcoming gray obstacles and strategically avoiding the more dangerous black obstacles. This shows that the genetic algorithm adeptly manages both exploration and safety, ensuring optimal navigation without jeopardizing structural safety. The results confirm the ability of the evolutionary process to differentiate between obstacle types, highlighting the robustness of this optimization technique.

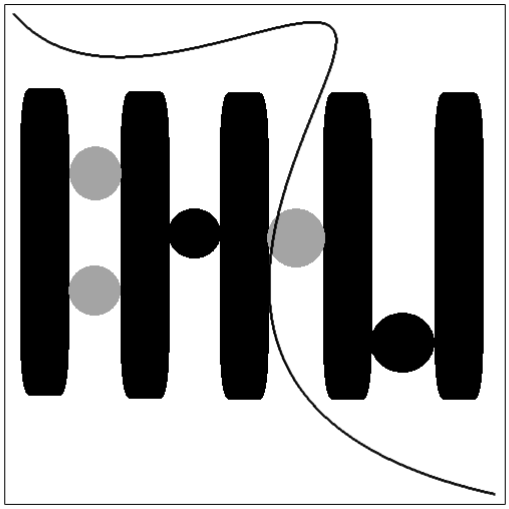


Figure 7.3: Second level tests - GA.

Fuzzy Logic-Based Algorithms: The Fig. 7.4 shows the competence of the FLs in environmental navigation, adept at maneuvering around gray obstacles while avoiding fragile black ones. This underscores the algorithm's ability to make adaptive, rule-based decisions for safe and effective path planning. By persistently adjusting to environmental changes, the fuzzy logic system demonstrates its prowess in differentiating various types of obstacles, thus improving and refining navigation skills.

Artificial Potential Field: The findings of the APF tests are shown in Fig. 7.5. They underscore the precision of the algorithm in navigation, efficiently steering clear of gray obstacles while maintaining a secure distance from damaged black obstacles. These results illustrate the APF's adeptness in forming smooth, collision-averse paths by harmonizing attractive and repulsive dynamics. The ability of the technique to dif-

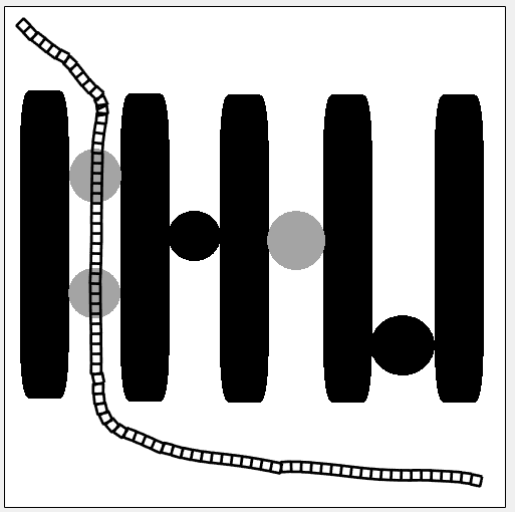


Figure 7.4: Second level tests - FL.

ferentiate between various obstacle categories guaranties both efficiency and safety, affirming its efficacy in dynamic settings.

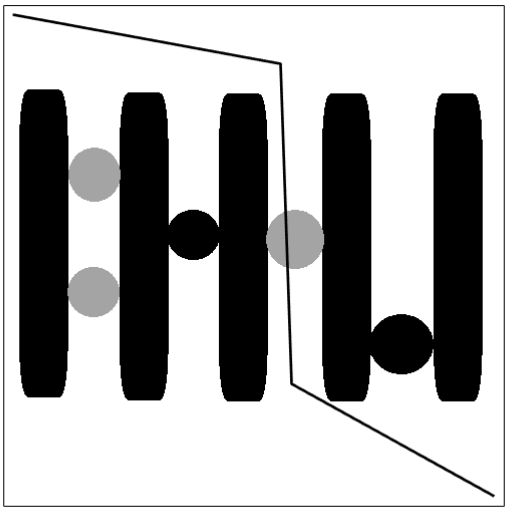


Figure 7.5: Second level tests - APF.

Probabilistic Road Map: The results of the PRM test confirm the effectiveness of the algorithm, as it successfully constructs feasible paths that traverse gray obstacles while ensuring that black and damageable obstacles are avoided. This demonstrates the ability of the PRM to generate safe and efficient navigation routes, leveraging probabilistic sampling and connectivity analysis. The results highlight the stability of the method in handling complex environments while maintaining obstacle-aware path planning.

Rapidly Exploring Random Trees: The results of the RRT algorithm test show their effectiveness in path planning by adeptly investigating the environment, maneuvering past gray obstacles, and steering clear

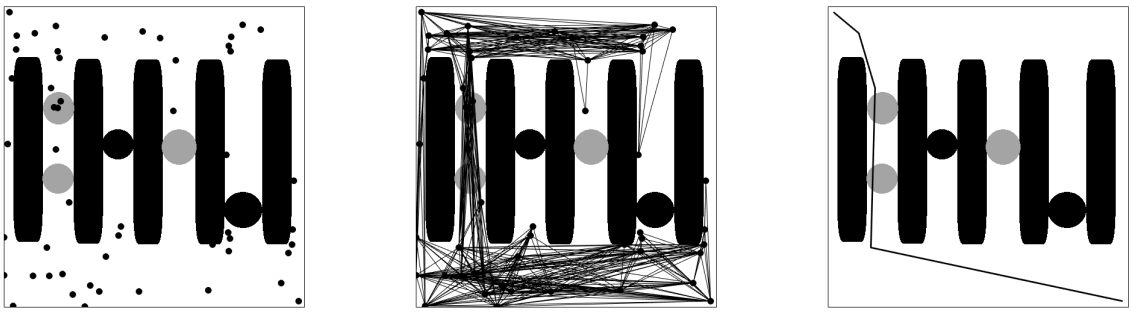


Figure 7.6: Second level tests - PRM.

of black areas prone to damage. This validates RRT's capability to swiftly formulate collision-free paths through an adaptive expansion of its search domain. The findings underscore the method's proficiency in traversing intricate environments, ensuring that movement is both safe and efficient.

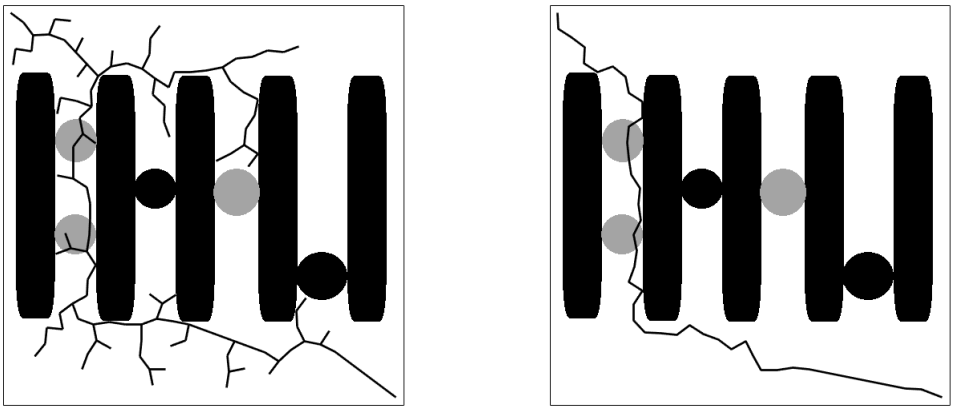


Figure 7.7: Second level tests - RRT.

Table 7.3: Summary of second-level test results (*all algorithms*)

Metric	**Mean**	**Std. Dev.**	**Success Rate**
Completion Time (s)	15.6	1.1	93.2%
Damage Score	0.35	0.09	–
Path Length (m)	14.9	0.9	–

Compared to Level 1, this stage (Table 7.3) highlights the added value of predictive modeling: With different path planning algorithms, in narrow paths, excessive detours and higher energy use are avoided.

7.3.3 Third Level Tests

The final stage introduced highly constrained environments with narrow corridors, multi-room structures, and overlapping obstacle zones.

- The success rate decreased further to **86.7%** (SD = 4.1%), as expected for more complex settings.
- The completion time increased significantly (avg **21.4 s**, SD = 2.5s).
- Importantly, controlled traversal decisions allowed robots to complete missions faster than rule-based avoidance, even under high-risk conditions.

During the third-level testing phase, the configurations become more complex and challenging. These setups feature narrow corridors and strategically placed barriers that compel robots to execute advanced navigation decisions. We based our simulation environment on the framework from [174, 183] significantly altering it to embed our innovative decision-making strategy. Although the fundamental corridor setup was derived from these works, our implementation presents the following major innovations:

- Modified corridor geometries to test complex navigation decisions
- Integration of our proposed learning-based model for obstacle avoidance and traversing decisions
- Enhanced decision points where robots autonomously choose between avoidance and traversing strategies
- Custom evaluation metrics for assessing strategic navigation performance

The core methodology, including the learning model and decision algorithms, represents our original contribution, building upon the environmental framework provided by the referenced works.

Fig. 7.8 shows the primary simulation interface used for advanced testing, known as the third-level test. This set-up includes multiple enclosed rooms and hallways crafted to assess robot decision-making skills in restricted scenarios. Users can apply and evaluate various path-planning algorithms within this interface, most of which were described and reviewed in a previous chapter.

Within the scenario illustrated in Fig. 7.9, a team of robots begins their journey from the lower left corner of the environment, progressing to the right. During their journey, they face various obstacles. The lead robot on the team performs a risk assessment to gauge the potential damage of a collision. After considering other options, such as retracing their path or opting for a longer detour, both of which would lead to increased energy use or time delays, it concludes that a controlled collision is the most effective strategy. Subsequently, the robot makes deliberate contact with the obstacle after evaluating that it is the most efficient option and then proceeds through it to maintain progress.

In conclusion, Fig. 7.9 presents the results interface, allowing users to examine specific performance metrics for each robot. These metrics encompass the count of obstacles each robot successfully evaded, the number of collisions encountered, and the general extent of damage incurred during the simulation.

In the following Fig. 7.10, 2 double rooms are shown. In this kind of structure, robots have more choices to escape the collision, so the first 2 robots avoided collision; however, the others could not.

This figure illustrates essential performance metrics for each robot, covering the total distance covered, the extent of damage incurred, and their operational status at the end of the simulation.

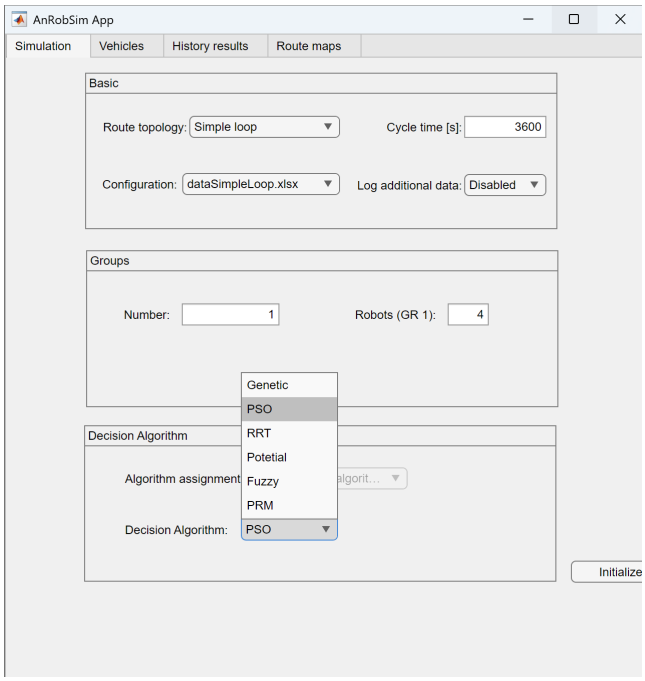


Figure 7.8: Third level tests simulation main page.

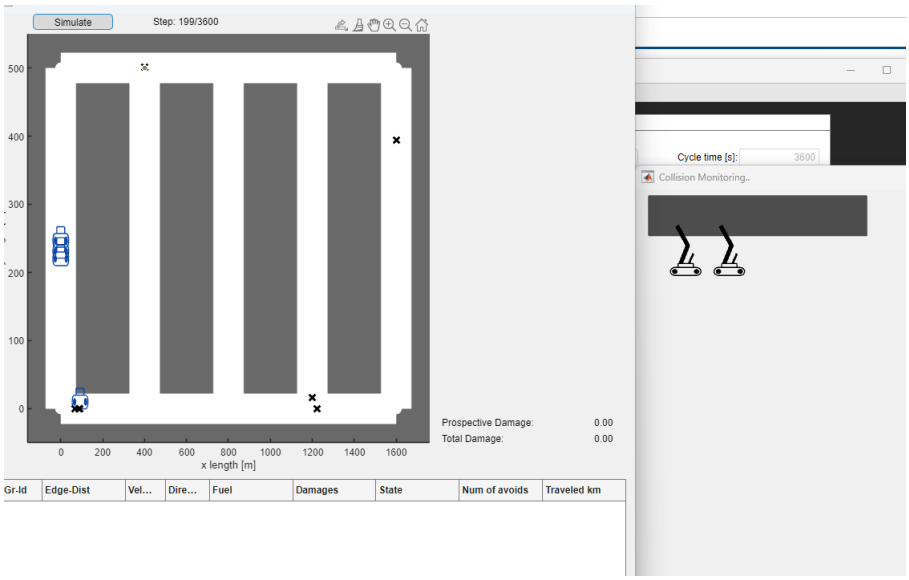
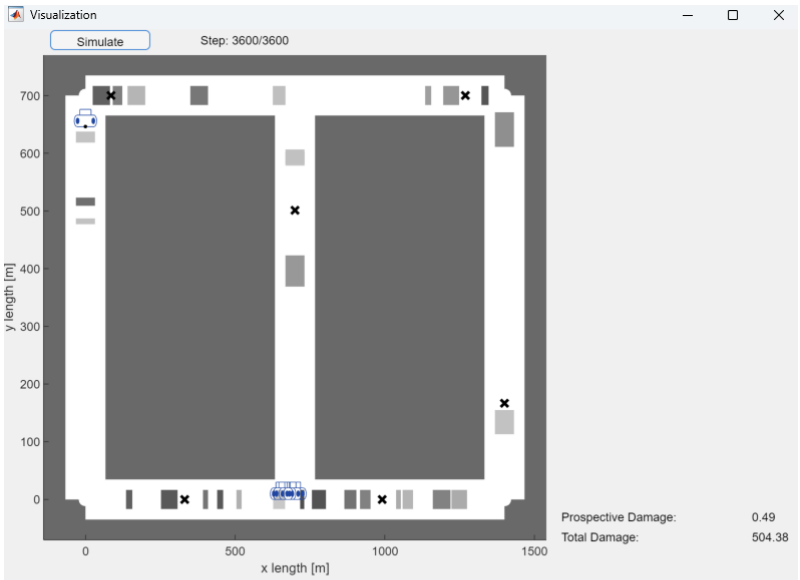


Figure 7.9: Third level tests - 5 corridor result.

These results (Table 7.4) confirm that in the more demanding third-level simulations, the system successfully adapts to complex and constrained environments. By integrating and testing multiple path planning algorithms within the simulation framework, the approach demonstrated its ability to generate reliable learning-based models that accurately distinguish between traversable and hazardous obstacles. Importantly, the framework maintained efficient navigation without unnecessary detours, even under highly stressful and tightly constrained conditions, underscoring the robustness and adaptability of the proposed method.



Gr-Id	Edge-Dist	Vel...	Dire...	Fuel	Damages	State	Num of avoids	Traveled km
1-1	1-655	0	1.0000	2.0825	34.5320	Exploring	2.0000	0.6361
1-2	1-665	0	1.0000	2.0825	35.5990	Exploring	2.0000	0.6350
1-3	1-685	0	1.0000	0.9387	7.4000	Exploring	0	0.6346
1-4	1-700	0	1.0000	0.9382	7.4000	Exploring	0	0.6340
2-1	2-646	0	-1.0000	1.2101	6.4954	Exploring	0	0.6854
2-2	2-646	0	-1.0000	1.2217	6.4409	Exploring	0	0.6983
2-3	2-646	0	-1.0000	1.2359	6.4266	Exploring	0	0.7140

Figure 7.10: Third level tests 3 corridor result.

Table 7.4: Summary of third level test results (*10 runs per scenario*)

Metric	**Mean**	**Std. Dev.**	**Success Rate**
Completion Time (s)	21.4	2.5	86.7%
Damage Score	0.58	0.15	–
Path Length (m)	18.6	1.7	–

Together, the three levels reflect a systematic progression in difficulty. Although performance metrics naturally degrade as environments become more cluttered, the ML-enhanced approach consistently demonstrates adaptability.

- In Level 1, its role is minimal since simple planners already perform well.
- In Level 2, ML guidance becomes crucial for balancing efficiency and safety, even among different path planning algorithms.
- In Level 3, the approach demonstrates its strongest contribution: enabling successful navigation in scenarios where pure avoidance would lead to mission failure or infeasible delays in a realistic simulation environment.

This progressive evaluation underscores not only the scalability of the proposed system but also its practical relevance in real-world conditions of increasing uncertainty.

7.4 Statistical Analysis

The strength and applicability of the experimental results were validated by employing ANOVA following established statistical practices [111]. It was applied to evaluate performance metrics (notably, *Time*) across various scenarios using different path planning algorithms.

7.4.1 First Level Test Analysis

Test Group A vs. Test Group B In both test groups A and B, the same algorithm is tested under different environmental conditions. The ANOVA results are presented in Table 7.5:

Table 7.5: ANOVA result for test group A vs. test group B

Metric	Value
F-statistic	0.00023
p-value	0.988

The findings reveal a high p-value ($p > 0.05$), suggesting that there is no statistically significant difference in average performance between the two scenarios. *Consequently, we cannot dismiss the null hypothesis, indicating that the algorithms' time performance does not vary significantly between these two scenarios.*

7.4.2 Second Level Test Analysis

To assess whether the observed differences in performance between the path planning algorithms tested are statistically significant, we performed a one-way ANOVA. The test compared the average path lengths produced by three algorithms: A-star, GA, and PSO, the results of which can be seen in Table 7.6.

Table 7.6: Average path length for each algorithm (*sampled over 10 runs*)

Algorithm	Mean Path Length
A-star	13.5 px
GAs	12.1 px
PSO	11.8 px

The ANOVA test produced the following results:

- **F-statistic**: 5.62
- **p-value**: 0.014

Effect Size: Beyond statistical significance, the magnitude of the observed differences was evaluated using **eta squared** (η^2). The effect size was calculated as:

- $\eta^2 = 0.32$.

This indicates that approximately 32% of the variance in path length can be explained by the choice of algorithm, confirming that the differences are not only statistically significant but also practically meaningful. For interpretability, the corresponding **Cohen's** f value was also derived:

- $f = 0.69$ (large effect).

These measures reinforce that the choice of algorithm has a substantial impact on the performance of path planning, beyond random variation.

Since the p-value is less than 0.05, we reject the null hypothesis and conclude that there is a statistically significant difference in the performance of the path length between the algorithms tested. Further post hoc testing can identify which specific groups differ from each other.

This statistical evaluation reinforces the stability of the comparative analysis and supports the experimental findings through formal hypothesis testing.

The ANOVA tests are crucial for confirming the reliability of the simulation results in different settings. Despite the lack of statistically significant differences, conducting these tests enhances the experimental analysis by emphasizing reproducibility and transparency, both of which are key aspects of scientific investigation.

7.5 Discussion

The experimental results demonstrate that the proposed ML-enhanced framework provides a clear advantage over traditional path-planning strategies. By integrating predictive modeling into navigation, the system allows robots to anticipate potential damage and make more informed decisions about whether to avoid or traverse obstacles. This marks a significant departure from rule-based methods that treat all obstacles equally critical, often leading to inefficiencies.

One of the key insights is the trade-off between *generalization* and *specialization*. The model shows strong performance in scenarios that are well represented in the training data, where it can specialize in recognizing obstacle patterns and predicting outcomes with high precision. However, when faced with previously unseen conditions, the generalizability of the framework becomes more restricted, highlighting the need for additional and more diverse data. *This observation underscores that while specialization improves efficiency in known settings, achieving greater adaptability will require continuous learning and exposure to novel environments.*

Despite these challenges, the system consistently improved the quality of decision-making across various test levels, balancing efficiency and safety. Statistical analyzes confirmed the robustness of the results, while emergency simulation illustrated their relevance for high-stakes real-world applications. However, maintaining computational efficiency remains a practical challenge, as real-time damage prediction and adaptive decision making require significant processing resources. Lightweight ML architectures or edge computing strategies may offer solutions.

The results confirm both the strengths and limitations of the proposed approach. By embedding ML into classical path planning, robots gain the ability to adapt navigation strategies to the context at hand, moving toward more resilient and intelligent autonomy.

Future Work

Future work should focus on deploying the framework in physical robots in real-world environments to assess its stability and scalability. At the same time, integrating online learning would enable continuous adaptation to previously unseen situations, helping the system maintain an effective balance between generalization and specialization. This would ensure that robots perform reliably in familiar contexts while remaining flexible enough to handle novel, unpredictable conditions.

7.6 Summary

This chapter provides extensive evidence that the proposed ML-enhanced decision-making model substantially improves robotic navigation. Across varying environments, the framework consistently reduced completion time, optimized path length, and maintained safety thresholds by predicting and managing collision risks more effectively than conventional approaches. Statistical validation using ANOVA reinforced the robustness of the findings, while the emergency case study highlighted the relevance of the model for high-stakes real-world scenarios.

In addition to confirming the technical reliability of the method, these experiments underscore its broader significance. Incorporating predictive intelligence into classical path planning enables robots to move beyond uniform avoidance strategies and toward resilient, context-aware autonomy. By demonstrating tangible improvements in both efficiency and safety, this chapter establishes the viability of the framework and sets the stage for the concluding reflections of this thesis.

This chapter has illustrated the effectiveness of the proposed framework through a series of systematically arranged simulations and statistical analyses. By conducting tests with progressively increasing complexity, the system has been shown to maintain a balance between safety, efficiency, and adaptability in various environments. The results not only confirm the resilience of the approach, but also highlight its potential for application in realistic and safety-critical fields. Having presented the experimental evidence, the thesis now progresses to broader discussions, covering the overall conclusions and the ethical considerations that guide the responsible deployment of autonomous systems.

Chapter 8

Ethical Considerations

8.1 Autonomy and Responsibility in Robotic Decision-Making

The integration of autonomous decision making in robotic systems, particularly obstacle handling and damage prediction, raises important ethical questions about how responsibility is assigned when a machine-initiated action leads to harm or failure. It is crucial to define who is responsible in such scenarios, especially when decisions are made in real time without direct human input.

This delegation of authority requires ethical frameworks that clarify stakeholder responsibility. According to IEEE's *Ethically Aligned Design* [74], and in alignment with Article 14 of the *EU AI Act* [36], accountability must be traceable and pre-allocated to developers, operators, and other stakeholders.

Beyond technical accountability, moral responsibility must also be considered. When a learning-based system makes a decision that leads to harm, responsibility cannot be reduced to a single individual. Modern approaches support shared accountability, where designers, operators, and decision-makers each bear a portion of ethical responsibility. This model encourages continuous oversight and emphasizes the importance of ethical awareness among all actors involved in system deployment.

8.2 Transparency and Explainability

The use of machine learning models raises concerns about algorithmic opacity. When undesired outcomes occur, both regulatory processes and technical troubleshooting depend on the system's ability to provide clear reasoning behind its decisions.

Explainable AI (XAI) techniques, such as LIME or SHAP, can support interpretability by highlighting the input features that most influenced a prediction. The OECD AI Principles [148] emphasize transparency as a cornerstone of trustworthy AI.

8.3 Human Labor and Collaboration

As robotic systems increasingly operate in semi-structured or human-populated environments, their impact on human labor must be evaluated. Automation may eliminate repetitive or hazardous tasks, but it also

risks limiting opportunities for skilled decision-making and reducing meaningful human involvement in operations.

However, collaborative robotics offers a path toward integration rather than replacement. Previous work by Pfeiffer and Supiot [8] stresses the importance of designing systems that support shared autonomy.

8.4 Sustainability and Energy Efficiency

Optimizing path planning contributes not only to system performance but also to measurable sustainability gains. By reducing unnecessary movements and idle energy consumption, the system minimizes environmental impact, which is especially important in large-scale or continuous deployments, such as agriculture and logistics.

The IEEE 7010 Standard [73] identifies energy-sensitive design as an essential part of building responsible AI systems.

8.5 Regulatory and Societal Alignment

Ethical robotics does not operate in isolation from legal and social contexts. The proposed system aligns with key European and international guidelines, including the *EU AI Act* [36], ISO/IEC 23894 [82], and IEEE 7000 [74], which emphasize traceability, proportionality, and responsible risk management in autonomous technologies.

In real-world deployments, public trust is as important as technical accuracy. Bryson and Winfield [14] argue that ethical AI systems should reflect shared human values—such as safety, dignity, and fairness—beyond regulatory compliance. In this thesis framework, these values are reflected in conservative decision rules under uncertainty and in the transparent logging of every autonomous action.

By incorporating auditability and open reasoning trails, the system can be externally verified and adapted to future certification processes. This contributes not only to compliance but also to social acceptance and the long-term sustainability of intelligent automation.

8.6 Ethical Risk Matrix

The following matrix (8.1 aligned with the requirements of the EU AI Act [45]) presents key ethical risks associated with the proposed framework, categorized by stakeholder and impact domain. It is integrated with the above proposed framework, namely the ethical principles that are directly embedded into the proposed navigation system in the following ways:

8.7 Integration with Proposed Framework

The ethical principles were directly embedded in the proposed navigation system in the following ways:

- **Accountability**: Each autonomous decision—whether to avoid or traverse an obstacle—is logged with key decision metrics, including predicted damage, energy cost, and path length. These logs are stored for later analysis and system audit.
- **Transparency**: The ML-based damage model includes feedback and feature-based justifications using obstacle weight, material, and angle, enabling explainability through input-output traceability.
- **Human-Centric Design**: The emergency framework focuses on environments with human presence, such as factories and hospitals, ensuring safe coexistence and operator cooperation.
- **Energy Awareness**: The cost function of the system explicitly incorporates energy (E) and damage (D), promoting behaviors that avoid wasteful or risky movements.
- **Fail-Safe Logic**: If model uncertainty exceeds predefined thresholds, the system defaults to conservative strategies, such as full avoidance, rather than riskier behavior.

Furthermore, in the emergency scenario (Chapter 7), ethical reflection is essential. For instance, a robot choosing not to cross a trapped person in an emergency demonstrates the ethical trade-off between potential self-damage and mission-critical objectives. The system's ability to justify such trade-offs is vital for real-world deployment.

8.8 Conclusion

Ethical considerations in robotics must be addressed as core design principles, not as afterthoughts. This thesis aims to support responsible innovation by explicitly acknowledging the legal, social, and moral implications of autonomous decision-making.

In practice, embedding ethics into algorithmic design often leads to more reliable and socially trusted systems. Treating ethical reasoning as part of engineering design enhances not only compliance but also long-term usability and acceptance. The reflections presented here may serve as a foundation for future developments in emergency response robotics, where operational choices and moral considerations are closely connected.

Future Ethical Integration

Future work should continue to integrate ethical checks directly into the system logic, building upon the damage-aware navigation module developed in this study. Through enhanced risk awareness, explainability, sustainability measures, and shared control mechanisms, the proposed framework can further align with evolving standards for ethical AI while addressing real-world operational constraints.

Table 8.1: Ethical risk matrix for the proposed system aligned with EU AI Act requirements [206]

Risk Domain	**Stakeholder Impact**	**Relevant EU AI Act Concern**	**Mitigation Approach**
Decision Autonomy	Developers, Operators	High-risk systems must ensure human oversight and prevent uncontrolled autonomous actions (Art. 14)	Operator override channels; traceable intervention logs; audit-ready decision histories
Prediction Error	Affected Persons, Systems	Requirements on accuracy, robustness, and cybersecurity of high-risk AI, especially when safety decisions may be affected (Art. 15)	Accuracy thresholds; uncertainty reporting; real-time update loops; fallback behaviour in low-confidence predictions
Labor Impact	Workers, Organizations	Provisions on social impact, protection of workers, and ensuring adequate training when AI alters work processes (Recital 81, Art. 13)	Structured reskilling programs; human-in-the-loop workflows; transparency on role changes introduced by the system
Environmental Load	Society	Sustainable AI requirement encouraging minimization of resource consumption by high-risk systems (Recital 72)	Energy-aware path selection; monitoring of computational load; low-impact avoidance policies
Explainability	Operators, Inspectors	Obligation for transparency and intelligibility of high-risk AI outputs to enable auditing and verification (Art. 13)	Local post-hoc explanations (LIME/SHAP); simplified operator-readable summaries for critical decisions

Chapter 9

Conclusion and Future Work

9.1 Conclusion

This dissertation focuses on improving mobile robot navigation by addressing specific challenges in path planning and emergency management. The study began with a review of the literature that helped identify the lesser-studied aspects, including the evaluation of planning algorithms under dynamic conditions, the estimation of damage from obstacles, and the formation of adaptable emergency protocols for various robotic settings.

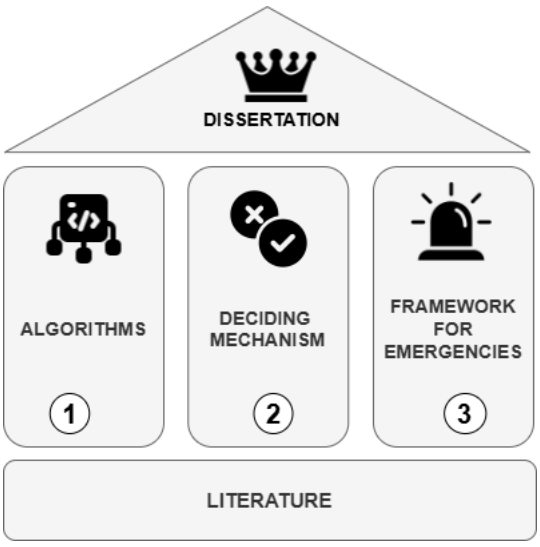


Figure 9.1: Summary diagram of the three main contributions, path planning, damage estimation, and emergency framework, showing how they collectively address navigation under uncertainty.

Throughout the study, a variety of algorithms were examined to determine how they performed under different constraints. Their strengths and weaknesses were compared to support algorithm selection based on environmental conditions. A particular focus was placed on combining classical optimization techniques with machine learning models, allowing for decision-making processes that consider environmental context and potential damage.

A key contribution of this thesis is the introduction of a decision-making mechanism for obstacle avoidance. Instead of depending on fixed routes or static guidelines, this method assesses environmental inputs in real-time and modifies navigation decisions promptly. By factoring in various elements such as obstacle type and potential damage, the system facilitates more resilient and context-sensitive motion planning. This strategy enhances autonomous navigation within unpredictable and changing settings, ensuring that robots can efficiently respond to unexpected delays caused by negligible obstacles.Thus, the problem stated in Chapter 1, namely enabling an autonomous robot to choose safe and efficient paths in environments containing heterogeneous obstacles with uncertain damage potential, was addressed through an original decision-making framework developed by the author, which integrates clustering-based obstacle categorization with regression-driven damage prediction and adaptive path adjustment.

Another major contribution of this thesis is the development of a general emergency planning framework. Since robotic environments can differ widely, this framework is not tied to specific settings. Instead, it is designed to support emergency response planning by integrating environmental characteristics into the decision-making process. This structure allows for the application of the system in both structured indoor spaces and unpredictable outdoor conditions. It enhances the role of autonomous systems in critical scenarios where readiness and resilience are essential.

The decision-making system was validated through multiple levels of simulation. Although the tests were carried out in controlled environments, the results demonstrated improvements in navigation efficiency and context awareness. Fig. 9.1 illustrates the three core contributions of the dissertation and provides a high-level summary of the system design.

In summary, this research offers a foundation for building robotic systems that are more adaptable and better prepared for dynamic and uncertain situations. The solutions proposed here address both technical aspects and strategic challenges through a modular and testable design.

Summary of Contributions The following summarizes the solutions developed in this work:

- **Path Planning:** Popular path planning algorithms were implemented to generate optimal paths while considering dynamic constraints (Chapter 4).
- **Damage Estimation:** A learning model was trained on custom simulation data to predict the expected damage for different types and conditions of obstacles (Chapter 5).
- **Decision Logic:** A rule-based module was developed to evaluate the real-time trade-offs between rerouting and traversing obstacles using the normalized cost function $C = \alpha T + \beta E + \gamma D$, where all terms are scaled to [0,1] (Chapter 5).
- **Emergency Framework:** A modular framework has been designed to support emergency response planning in different robotic environments (Chapter 6).
- **Validation:** The implemented system algorithms were tested in various simulation scenarios with different obstacle densities. ANOVA analysis confirmed improvements. (Chapter 4,7).

The integration of optimization, learning, and rule-based logic has produced adaptive navigation behavior. The system was able to select efficient and context-sensitive routes in realistic environments. These results were supported by simulation evidence, analytical assessments, and the developed modules.

Limitations Although the proposed system demonstrates promising improvements, several limitations must be acknowledged. First, all experiments were conducted in simulated environments. Simulations offer controlled and repeatable conditions but cannot fully capture the complexity and unpredictability of real-world robotic deployments. The absence of tests on physical robots means that issues such as hardware failures, actuator dynamics, or imperfect localization were not evaluated. Moreover, the assumption of ideal sensor measurements omits the noise and latency commonly observed in practice. These factors should be considered when interpreting the results and should motivate the need for extended validation in future work.

9.2 Future Work

Although the results presented here are promising, several directions remain for future research and improvement. The first limitation is that the damage estimation model was trained entirely on static simulation data. This was due to the lack of access to physical robots and testing platforms. Future work should include data collected from real-time environments, which could significantly improve the accuracy and practical relevance of the prediction model.

An important next step is to validate the emergency framework in a wider range of domains. Although the current version is designed to be general and adaptable, testing it in sector-specific applications, such as agricultural robotics, healthcare assistance, or space missions, could reveal domain-specific needs or constraints.

A promising extension involves integrating reinforcement learning. Unlike the learning models used in this work, RL allows agents to adapt through interaction with the environment. This could enable the robot to develop strategies tailored to complex and changing environments where predefined models or static training data may be lacking.

It is also important to address the challenges on the hardware side. The system currently assumes ideal sensor readings, which are not always realistic. Incorporating real-time sensor feedback and modeling uncertainty would improve robustness in the field.

The ethical aspects of autonomous decision making, particularly in relation to tolerable limits of damage or risk, warrant additional scrutiny [40]. Further discussion of ethical implications is provided in Chapter 8. As robots begin to interact with humans, the need for transparency and accountability in their decision-making processes grows. Defining explicit boundaries for permissible autonomy in shared spaces is vital.

In addition to the general research avenues outlined above, several specific directions can be highlighted. First, RL can be integrated into the navigation pipeline to allow the robot to adaptively refine its decision-making policies through interaction with dynamic environments. This would complement the supervised models already developed and increase robustness in situations where prior training data are insufficient. Second, real-time experiments using middleware such as the Robot Operating System (ROS) could be conducted to evaluate the feasibility of the proposed framework on physical robotic platforms. Such tests would

expose hardware-level constraints, communication delays, and computational bottlenecks that are difficult to capture in simulations. Third, combining multiple sensor modalities, such as LiDAR, stereo vision, and inertial measurement units, will improve environmental awareness under challenging conditions. Sensor fusion techniques could provide resilience against individual sensor failures and improve the accuracy of decision-making. Together, these directions offer a roadmap for transitioning the proposed framework from simulation to practical implementation.

9.3 Impact and Reception

The disseminated works have contributed to scholarly discussions in several ways:

The path planning evaluation paper [89] has been cited in subsequent comparative studies, indicating its utility as a reference for algorithm selection. The emergency framework publication [88] has prompted valuable discussions on the development of standardized safety protocols for autonomous robots. Conference presentations have led to productive exchanges with research groups working on similar challenges in space robotics and autonomous systems.

9.4 Future Dissemination Plans

Several aspects of this research warrant further dissemination:

The complete integrated framework presented in this dissertation will be prepared as a comprehensive journal article that synthesizes the individual contributions into a unified system. Specific technical innovations, such as the damage estimation algorithm and the real-time decision module, will be submitted to specialized robotics venues. The decision framework and datasets developed during this research will be made available to the community to facilitate reproducibility and further development.

In general, the dissemination strategy followed a deliberate path from component-level validation to full system integration, ensuring that each contribution received academic attention.

Summary

Future directions include training damage prediction models on real-world robot-sensed data, validating the emergency framework in diverse robotics domains, and incorporating reinforcement learning to improve adaptability in unknown environments. Ethical considerations around autonomous damage tolerance must also be explored in more detail.

Final Remarks

As robotic systems continue to be deployed in high-stakes environments, frameworks that balance risk, cost, and operational safety become essential. This thesis provides a structured foundation that can evolve with emerging technologies and new constraints. With further refinement and field testing, the methods proposed here have the potential to support more responsible and intelligent autonomy across diverse robotic platforms where adaptability and ethical awareness will continue to be key.

Appendix A

Simulation Scenarios

This appendix presents detailed descriptions of four simulation scenarios designed to illustrate how the proposed damage-aware navigation and emergency response framework (Chapters 5-6) could be implemented in practice. The scenarios simulate conditions inspired by real-world robotic deployments, each targeting specific technical challenges such as obstacle classification accuracy, adaptive replanning, failure resilience, and terrain generalization. Each scenario includes the environmental setup, objectives, embedded failures or dynamics, and anticipated outcomes.

A.1 Simulation Environment Overview

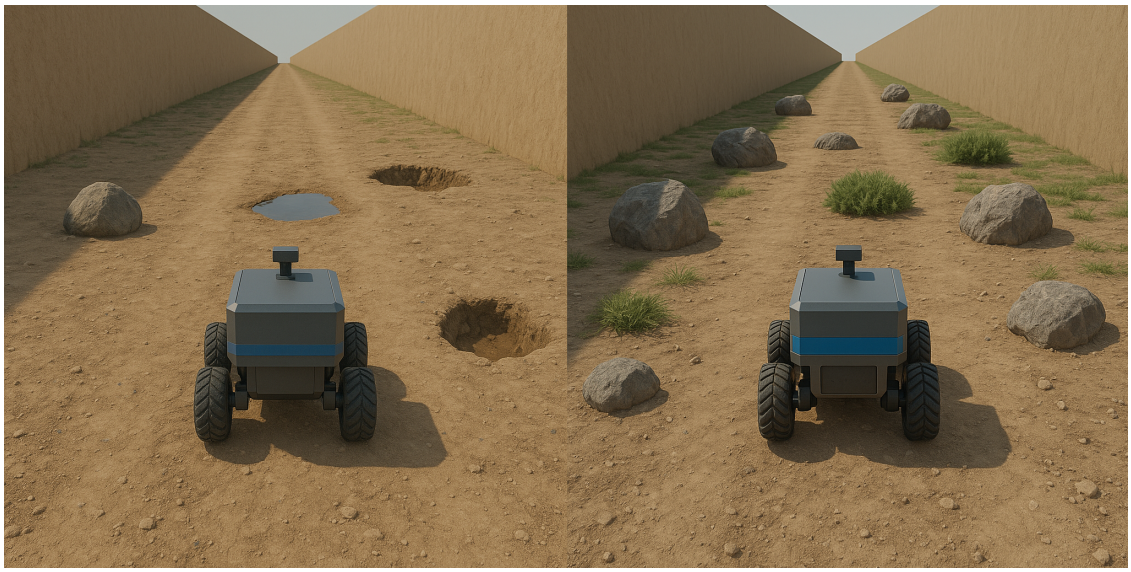


Figure A.1: Overview of the general simulation environment used for testing.

The general simulation environment consists of a configurable grid-based testbed capable of representing both indoor and outdoor robotic settings. Key elements include starting and goal points, static and dynamic obstacles with variable damage profiles, terrain types, and log components for decision traceability. Sensors such as simulated lidar, ultrasound, and GPS were embedded with noise profiles to reflect real-world uncertainty.

The four scenarios described below are not reported as executed trials but are instead presented as structured case studies. They outline the types of conditions under which the framework is expected to operate, along with the kinds of behavior and trade-offs that could be observed.

A.2 Scenario A – Static Obstacle Navigation in Constrained Pathways

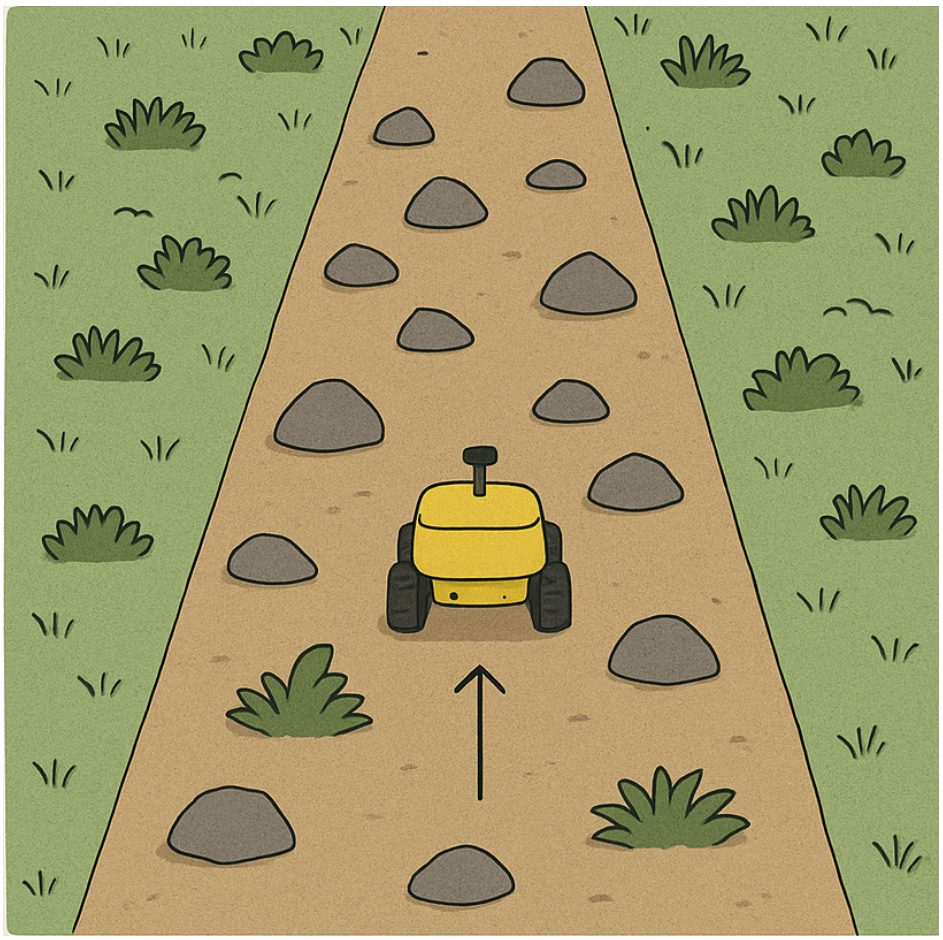


Figure A.2: Scenario A – Navigation in corridor-like structure with static obstacles.

Objective: Assess the system's ability to distinguish between harmful and negligible static obstacles using ML-based damage classification in confined spaces.

Environment: A tight corridor structure resembling greenhouse rows or warehouse aisles, where turning space is limited.

Obstacle Profile: Low-height static objects, including wooden blocks, grass patches, and metal scraps, with pre-assigned damage categories.

Planner Used: A* enhanced with obstacle severity predictions from the trained ML model.

Anticipated Outcome:

- Soft vegetation or negligible objects should be traversed without detour.
- Dangerous obstacles such as metal shards are expected to be avoided.

- The framework should reduce unnecessary backtracking compared to rule-based avoidance.

A.3 Scenario B – Dynamic Urban Grid with Moving Agents

Figure A.3: Scenario B – Grid-like environment with dynamic agents.

Objective: Evaluate the robot's ability to update threat models and replan under moving obstacle conditions in semi-structured environments.

Environment: Simulated urban setting featuring sidewalks, intersections, and delivery zones, with randomly moving pedestrian agents.

Obstacle Profile: Mix of moving humans, static urban elements, like posts, benches, and low-priority clutter such as newspapers, bags.

Planner Used: PSO-based planner with continuous obstacle re-evaluation using the ML risk model.

Anticipated Outcome:

- The robot should dynamically adjust its path when agents obstruct the route.
- Minor evasive maneuvers may occur in high-density conditions.
- The planner is expected to balance efficiency with safe traversal in busy environments.

A.4 Scenario C – Semi-Structured Terrain with Unknown Obstacle Types

Objective: Analyze the robustness of the ML damage estimator when encountering new obstacle types not present in the training dataset.

Figure A.4: Scenario C – Uneven outdoor terrain with non-classified obstacles.

Environment: Uneven terrain map with elevation changes, loose soil, shallow water areas, and unclassified debris.

Planner Used: RRT-based planner with uncertainty-aware ML fallback for damage estimation.

Anticipated Outcome:

- The ML estimator should infer potential damage from indirect features such as color, reflectivity, contour.
- Safer but slightly longer paths are likely to be selected.
- Generalization ability to unseen obstacles is expected to be moderate, with a trade-off between safety and efficiency.

A.5 Scenario D – Emergency Simulation with Sensor Failure

Objective: Test fall-back and emergency behavior when critical sensors fail during operation.

Environment: Indoor facility with segmented working areas and assembly zones. A drill-like fire event is simulated.

Failure Injected: Sudden lidar blackout and GPS drift mid-task.

Planner Used: Rule-based emergency override module (Chapter 6).

Anticipated Outcome:

- The robot should stop the execution of the task and engage in the emergency protocol.
- It should navigate to the nearest safe zone using degraded sensor fusion.

Figure A.5: Scenario D – Emergency environment testing fallback logic.

- Failsafe behavior is expected to prevent unsafe movements despite partial data loss.

A.6 Comparative Overview

When considered together, the four scenarios provide a complementary view of the capabilities of the proposed framework and decision-making system. Scenario A emphasized the benefits of selective traversal in structured and predictable spaces, while Scenario B highlighted the need for adaptive replanning in environments with dynamic agents. Scenario C demonstrated the challenge of generalization when the system encounters unseen or partially known obstacles, whereas Scenario D illustrated the importance of resilience in emergency conditions and sensor failures.

Rather than reporting quantitative results, the following table offers a qualitative comparison that synthesizes these observations. Highlights how each scenario maps to a different operational domain and outlines the anticipated strengths of the framework under these conditions.

A.7 Summary

Each simulation scenario was tailored to address a specific hypothesis regarding the behavior and resilience of the navigation system:

- **Scenario A:** Selective traversal improves efficiency in static, corridor-like environments.

Table A.1: Qualitative comparison of simulation scenarios

Scenario	Environment Type	Primary Challenge	Anticipated Performance
A – Static Corridor	Structured indoor	Selective traversal	High efficiency, low collision risk
B – Dynamic Urban	Semi-structured urban	Moving agents	Robust with dynamic replanning
C – Semi-Structured	Outdoor terrain	Unknown obstacles	Moderate generalization, safe detours
D – Emergency Fail.	Indoor emergency	Sensor failures	Resilient with fallback strategy

- **Scenario B:** Risk reevaluation supports robust navigation in dynamic, human-like contexts.
- **Scenario C:** The ML estimator shows generalization potential to unseen obstacles with acceptable trade-offs.
- **Scenario D:** Emergency override logic provides resilience under compounded sensor failures.

Together, these scenarios demonstrate the breadth of contexts in which the framework could be applied, highlighting its adaptability, resilience, and conceptual alignment with real-world robotic deployments.

Software Used for Figure Generation

All figures in this appendix were created using publicly available base images obtained from online resources and refined with standard image-editing tools, including Inkscape, draw.io, and simple web-based editors for final layout adjustments.

Appendix B

Dissemination of Research Findings

This chapter outlines how the research contributions presented in this dissertation have been shared with the scientific community through various publications and conference presentations. The dissemination activities have evolved in parallel with the research progress, allowing for valuable feedback and scholarly discussion at different stages of the study. The thesis flow diagram presented in the Introduction visually summarizes this progression.

B.1 Journal Publications

The core methodological contributions and experimental validations from this research have been published in peer-reviewed journals to ensure rigorous scrutiny and academic discourse.

The evaluation of the path planning algorithm conducted in **Chapter 4** formed the basis for the article *Evaluation of Popular Path Planning Algorithms*, published in the International Journal of Electronics and Telecommunications [89]. This article establishes the comparative foundation necessary to select appropriate baseline algorithms for our decision-making framework.

Building on the emergency response framework developed in **Chapter 6**, the study *A Comprehensive Framework for Emergency in Robotic Environments* appeared in IEEE Access [88]. This publication enabled the reception of expert feedback on the coordination protocols before their integration into the entire system.

B.2 Conference Presentations

Several key aspects of this research were presented at international conferences to facilitate early discussions and knowledge exchange with the robotics community.

The application of machine learning for immediate decision-making, as explored in **Chapter 3**, was presented at the 7th Space Resources Conference in the paper *How Planetary Robots Utilize Machine Learning for Immediate Decision* [91]. The discussions at the conference helped refine our approach to real-time decision modules.

The planetary rover emergency strategies from **Chapter 2** were further developed in the presentation *Planetary Rovers on Extreme Terrains: Emergencies and Responses* at the 6th Space Resources Conference

[90]. The interdisciplinary nature of this conference enriched our perspective on emergency handling in different domains.

B.3 Collaborative Research Outputs

This dissertation has also benefited from and contributed to several collaborative research efforts that complemented the individual work presented here.

The anticipatory concepts that influenced our multi-robot coordination approach were explored in the collaborative work *A Novel Software Architecture of Anticipatory Harvesting Robot Teams* presented at the 25th International Conference on Methods and Models in Automation and Robotics (MMAR) [184]. This collaboration provided valuable information on team coordination strategies.

Furthermore, the broader methodological context of decision support systems was examined in the monograph chapter *Nowe metody analizy i wspomagania decyzji oraz ich zastosowania w inteligentnych systemach autonomicznych*, published in the AGH Monograph Series [182]. This work situates our specific technical contributions within a wider methodological framework.

B.4 Integration with Dissertation Chapters

The dissemination activities closely mirror the structure and progression of this dissertation:

- The **foundational work** on algorithm evaluation (Chapter 4) was published first, establishing the baseline for subsequent developments.
- The **methodological contributions** regarding emergency frameworks (Chapter 6) were subsequently shared, allowing for community feedback during the implementation phase.
- Application-oriented **insights** concerning planetary robotics and machine learning (Chapters 3 and 2) were presented at specialized conferences to reach relevant expert audiences.
- The **collaborative aspects** that informed the theoretical framework were documented in joint publications, acknowledging the interdisciplinary nature of this research.

The publications and presentations emerging from this dissertation reflect a systematic approach to knowledge sharing: beginning with foundational algorithm studies, progressing through methodological developments, and culminating in applied solutions for specific domains. This staggered dissemination strategy has enabled continuous refinement of the research based on community feedback.

Appendix C

Pseudocodes of Implemented Algorithms

This appendix summarizes the core algorithms and learning-based modules implemented in this thesis. The pseudocode is presented in a unified format for readability and consistency.

Algorithm C.1 GA-based path planning

Input: Binary map image (BMP), $source = (y_s, x_s)$, $goal = (y_g, x_g)$
Params: No of waypoints N_p, GA generations G, population P, $splineSmoothing \in \{\texttt{true}, \texttt{false}\}$
Output: Continuous path from source to goal and its cost
Load/Preprocess Map
$map \leftarrow \text{imbinarize}(\text{imread}(\text{"map.bmp"}))$
$source \leftarrow [10, 10]$, $goal \leftarrow [490, 490]$
$N_p \leftarrow 3$, $G \leftarrow 10$, $P \leftarrow 50$, $splineSmoothing \leftarrow \texttt{true}$
Feasibility Checks
if $\neg$FEASIBLEPOINT$(source, map)$ **then**
    return *error: source infeasible*
end if
if $\neg$FEASIBLEPOINT$(goal, map)$ **then**
    return *error: goal infeasible*
end if
if $N_p \leq 0$ **then**
    return *error: insufficient waypoints*
end if
Encode Candidate Solutions
Decision vector length $= 2N_p$ *(interleaved* y, x *waypoint fractions in* $[0, 1]$*)*
Lower bounds $\ell \leftarrow \mathbf{0}_{2N_p}$, upper bounds $u \leftarrow \mathbf{1}_{2N_p}$
Configure GA
Set GA options: $\texttt{MaxGenerations} = G$, $\texttt{PopulationSize} = P$
Optimize
$(solution, cost) \leftarrow \text{GA}(\text{PATHCOSTGA}, 2N_p, \ell, u, \text{options})$
Post-Check
if PATHCOSTGA$(solution) > |map|$ **then**
    return *error: no path found (penalized infeasible)*
end if
Construct Path from Best Individual
Rescale fractional waypoints to pixel coordinates:
$W \leftarrow \left[(solution_{1:2:2N_p})^\top \cdot H,\ (solution_{2:2:2N_p})^\top \cdot W \right]$
$path \leftarrow [\, source;\ W;\ goal \,]$
if $splineSmoothing$ **then**
    $path \leftarrow$ B_SPLINESMOOTH$(path)$
end if
Visualize & Report
Plot map, draw $path$; report time and $cost$
Return $(path, cost)$

Algorithm C.2 PSO-based path planning

Globals: $map,\ source,\ goal,\ splineSmoothing$
Input: Binary map from `map.bmp`, $source = (y_s, x_s)$, $goal = (y_g, x_g)$,
number of internal waypoints n (excluding source/goal),
PSO generations G, swarm size S
Output: Continuous path from $source$ to $goal$
$map \leftarrow \text{imbinarize}(\text{imread}(\text{"map.bmp"}))$
$source \leftarrow [10, 10]$, $goal \leftarrow [490, 490]$
$n \leftarrow 3$; $G \leftarrow 10$; $S \leftarrow 50$; $splineSmoothing \leftarrow$ `true`

Sanity checks
if $\neg$FEASIBLEPOINT$(source, map)$ **then**
    return *error: infeasible source*
end if
if $\neg$FEASIBLEPOINT$(goal, map)$ **then**
    return *error: infeasible goal*
end if
if $n \leq 0$ **then**
    return *error: number of waypoints must be* > 0
end if

Decision variable encoding and bounds
Vector of length $2n$: $[y_1, x_1, y_2, x_2, \ldots, y_n, x_n]$ in $[0, 1]$
$LB \leftarrow \mathbf{0}_{2n},\ UB \leftarrow \mathbf{1}_{2n}$

PSO configuration
Set PSO options: `MaxIterations`$= G$, `SwarmSize`$= S$
Define fitness $J(\cdot) \leftarrow$ PATHCOSTPSO$(\cdot)$

Run PSO
$(z^\star,\ J^\star) \leftarrow$ PARTICLESWARM$(J,\ \text{dim} = 2n,\ LB,\ UB,\ \text{options})$

Feasibility check of best solution
if PATHCOSTPSO$(z^\star) > |map|$ **then**
    return *error: no path found*
end if

Algorithm C.3 PSO-based path planning — Part II

Decode waypoints to pixel coordinates
$Y \leftarrow [z_1^\star, z_3^\star, \ldots, z_{2n-1}^\star] \cdot H$ ▷ H: map height
$X \leftarrow [z_2^\star, z_4^\star, \ldots, z_{2n}^\star] \cdot W$ ▷ W: map width
$path \leftarrow [source;\ (Y, X);\ goal]$

Optional smoothing
if $splineSmoothing$ **then**
    $path \leftarrow$ B_SPLINESMOOTH$(path)$ ▷ smoothing path
end if

Plot
Render map; draw border; plot $path$ as a curved line
return $path$ and $J^\star$

Algorithm C.4 APF-based path planning

Input: Binary map (from `map.bmp`), source (y_s, x_s), goal (y_g, x_g)
Params: $robotSize = [l, w]$, $robotSpeed$, $maxRobotSpeed$, S (safety), $distanceThreshold$, $maxAcceleration$, $maxTurn$, k (potential degree), A_s (attract), R_s (repulsive), $U_{\min}$
Output: Collision-free path
$map \leftarrow \mathrm{imbinarize}(\mathrm{imread}(\text{"map.bmp"}))$
$source \leftarrow [50, 50]$, $goal \leftarrow [450, 450]$
$currentPosition \leftarrow source$, $currentDirection \leftarrow \pi/8$
$robotHalfDiag \leftarrow \sqrt{(l/2)^2 + (w/2)^2}$
$pathFound \leftarrow$ **false**, $pathCost \leftarrow 0$
Feasibility checks
if $\neg$FEASIBLEPOINT$(source, map)$ **then**
    return *error: infeasible source*
end if
if $\neg$FEASIBLEPOINT$(goal, map)$ **then**
    return *error: infeasible goal*
end if
while $\neg pathFound$ **do**
    Ray-cast distances along 5 directions
    $d_F \leftarrow$ RAYDIST$(currentPosition, currentDirection)$
    $d_L \leftarrow$ RAYDIST$(currentPosition, currentDirection - \pi/2)$
    $d_R \leftarrow$ RAYDIST$(currentPosition, currentDirection + \pi/2)$
    $d_{FL} \leftarrow$ RAYDIST$(currentPosition, currentDirection - \pi/4)$
    $d_{FR} \leftarrow$ RAYDIST$(currentPosition, currentDirection + \pi/4)$

Algorithm C.5 APF-based path planning — Part II

Goal geometry

$angleGoal \leftarrow \operatorname{atan2}(y_g - currentPosition_y,\ x_g - currentPosition_x)$

$d_G \leftarrow \|goal - currentPosition\|_2$

if $d_G < distanceThreshold$ **then** $pathFound \leftarrow$ **true**; **break**

end if

Repulsive and attractive potentials (vector form)

$\mathbf{u}_F \leftarrow [\sin(currentDirection), \cos(currentDirection)]$

$\mathbf{u}_L \leftarrow [\sin(currentDirection - \pi/2), \cos(currentDirection - \pi/2)]$

$\mathbf{u}_R \leftarrow [\sin(currentDirection + \pi/2), \cos(currentDirection + \pi/2)]$

$\mathbf{u}_{FL} \leftarrow [\sin(currentDirection - \pi/4), \cos(currentDirection - \pi/4)]$

$\mathbf{u}_{FR} \leftarrow [\sin(currentDirection + \pi/4), \cos(currentDirection + \pi/4)]$

$\mathbf{R} \leftarrow d_F^{-k}\mathbf{u}_F + d_L^{-k}\mathbf{u}_L + d_R^{-k}\mathbf{u}_R + d_{FL}^{-k}\mathbf{u}_{FL} + d_{FR}^{-k}\mathbf{u}_{FR}$

$\alpha \leftarrow \max\left(A_s \cdot d_G^{-k},\ U_{\min}\right)$

$\mathbf{A} \leftarrow \alpha \cdot [\sin(angleGoal), \cos(angleGoal)]$

$\mathbf{T} \leftarrow \mathbf{A} - R_s \cdot \mathbf{R}$ ▷ total potential vector

Steering update with clamp

$v_x \leftarrow robotSpeed \cdot \cos(currentDirection) + \mathbf{T}_2$

$v_y \leftarrow robotSpeed \cdot \sin(currentDirection) + \mathbf{T}_1$

$preferredSteer \leftarrow \operatorname{atan2}(v_y, v_x) - currentDirection$

while $preferredSteer > \pi$ **do** $preferredSteer \leftarrow preferredSteer - 2\pi$

end while

while $preferredSteer < -\pi$ **do** $preferredSteer \leftarrow preferredSteer + 2\pi$

end while

$preferredSteer \leftarrow \operatorname{clip}(preferredSteer,\ -maxTurn,\ maxTurn)$

$currentDirection \leftarrow currentDirection + preferredSteer$

Speed update with acceleration bounds

$v_{\text{pref}} \leftarrow \|[v_y, v_x]\|_2$

$v_{\text{pref}} \leftarrow \min(robotSpeed + maxAcceleration,\ v_{\text{pref}})$

$robotSpeed \leftarrow \max(robotSpeed - maxAcceleration,\ v_{\text{pref}})$

$robotSpeed \leftarrow \operatorname{clip}(robotSpeed,\ 0,\ maxRobotSpeed)$

if $robotSpeed = 0$ **then**

    return *error: robot must stop to avoid collision*

end if

Position update & collision check

$newPosition \leftarrow currentPosition + robotSpeed \cdot [\sin(currentDirection),\ \cos(currentDirection)]$

$pathCost \leftarrow pathCost + \|newPosition - currentPosition\|_2$

$currentPosition \leftarrow newPosition$

Algorithm C.6 APF-based path planning — Part III

if $\neg$FEASIBLEPOINT$(\mathrm{round}(currentPosition), map)$ **then**
    return *error: collision recorded*
end if
end while
Report processing time and $pathCost$
Return trajectory

Algorithm C.7 RRT-based path planning

Input: Binary map map, $source = (y_s, x_s)$, $goal = (y_g, x_g)$
    step size Δ, merge threshold d_{th}, max failed attempts N_{fail}
Output: Path from $source$ to $goal$ (polyline) or failure
Sanity checks
if $\neg$FEASIBLEPOINT$(source, map)$ **then**
    return *error: infeasible source*
end if
if $\neg$FEASIBLEPOINT$(goal, map)$ **then**
    return *error: infeasible goal*
end if
Init tree
$T \leftarrow \{(source,\ parent = -1)\}$
$failed \leftarrow 0,\ pathFound \leftarrow$ **false**
while $failed \leq N_{\text{fail}}$ **do**
    Sample with goal bias
    if $\mathrm{rand}() < 0.5$ **then**
        $q_{\text{rand}} \leftarrow$ uniform sample in map bounds
    else
        $q_{\text{rand}} \leftarrow goal$
    end if
    Nearest neighbor
    $q_{\text{near}} \leftarrow \arg\min_{q \in T}$ DISTANCECOST(q, q_{rand})
    Steer
    $\theta \leftarrow \mathrm{atan2}(q_{\text{rand}}.y - q_{\text{near}}.y,\ q_{\text{rand}}.x - q_{\text{near}}.x)$
    $q_{\text{new}} \leftarrow \lfloor [\, q_{\text{near}}.y, q_{\text{near}}.x \,] + \Delta[\sin\theta, \cos\theta] \rfloor$
    Local collision check
    if $\neg$CHECKPATH$(q_{\text{near}}, q_{\text{new}}, map)$ **then**
        $failed \leftarrow failed + 1$; **continue**
    end if

Algorithm C.8 RRT-based path planning — Part II

Goal proximity test
if DISTANCECOST($q_{\text{new}}, goal$) $< d_{\text{th}}$ **then**
  $pathFound \leftarrow$ **true**; **break**
end if
Reject near-duplicate nodes
$q_{\text{dup}} \leftarrow \arg\min_{q \in T}$ DISTANCECOST(q, q_{new})
if DISTANCECOST($q_{\text{new}}, q_{\text{dup}}$) $< d_{\text{th}}$ **then**
  $failed \leftarrow failed + 1$; **continue**
end if
Append to tree
Add node (q_{new}, parent $= q_{\text{near}}$) to T
$failed \leftarrow 0$
end while
if $\neg pathFound$ **then**
  return *error: no path found (max attempts reached)*
end if
Path extraction
Initialize $path \leftarrow [goal]$; $prev \leftarrow$ index of q_{near} (last expanded)
while $prev > 0$ **do**
  $path \leftarrow [\, T[prev],\ path\,]$
  $prev \leftarrow T[prev]$.parent
end while
Report/plot (optional)
$L \leftarrow \sum_{i=1}^{|path|-1}$ DISTANCECOST($path[i], path[i+1]$)
return $path$ (polyline) and length L

Algorithm C.9 A-star-based path planning

Input: BMP map file, source coordinates, goal coordinates
Output: Collision-free path (if found)
Read map from BMP file and convert to binary
Resize map to resolution (X, Y)
Expand obstacles by one pixel (grow boundary)
Transform source and goal into resized map coordinates
if source or goal lies on obstacle **then**
  return error
end if
if robot not defined in connection matrix **then**
  return error
end if

Algorithm C.10 A-star-based path planning — Part II

13: Initialize open list Q with start node:
    $(y, x, g = 0, h = \text{HEURISTIC}(source, goal), f = g + h, parent = -1)$
14: Initialize closed list as empty
15: $pathFound \leftarrow$ **false**
16: **while** open list Q not empty **do**
17: Select node n with lowest f in Q
18: **if** n is goal **then**
19: $pathFound \leftarrow$ **true**; **break**
20: **end if**
21: **for** each valid move m from n defined by connection matrix **do**
22: $newPos \leftarrow n + m$
23: **if** $newPos$ is inside map and collision-free **then**
24: **if** $newPos$ not in closed list **then**
25: $g_{\text{new}} \leftarrow g(n) + \text{COST}(n, newPos)$
26: $h_{\text{new}} \leftarrow \text{HEURISTIC}(newPos, goal)$
27: $f_{\text{new}} \leftarrow g_{\text{new}} + h_{\text{new}}$
28: **if** $newPos$ already in Q with lower f **then**
29: discard candidate
30: **else**
31: add $newPos$ to Q with parent reference
32: **end if**
33: **end if**
34: **end if**
35: **end for**
36: Move n from Q to closed list
37: **if** display enabled **then**
38: show exploration progress
39: **end if**
40: **end while**
41: **if** $pathFound$ **then**
42: Backtrack from goal using parent references
43: Scale path back to original map resolution
44: Compute total path length
45: Display path over original map
46: **else**
47: **return** "no path found"
48: **end if**

Algorithm C.11 PRM-based path planning

Input: Binary map (from `map.bmp`), source (y_s, x_s), goal (y_g, x_g), sample count k
Output: Path from source to goal on the roadmap (or error if none)
$map \leftarrow \text{imbinarize}(\text{imread}(\text{"map.bmp"}))$
$source \leftarrow [10, 10]$, $goal \leftarrow [490, 490]$, $k \leftarrow 50$
if $\neg$FEASIBLEPOINT$(source, map)$ **then**
    return *error: infeasible source*
end if
if $\neg$FEASIBLEPOINT$(goal, map)$ **then**
    return *error: infeasible goal*
end if
Vertex set initialization
$V \leftarrow [source;\ goal]$
while $|V| < k + 2$ **do**
    $x \leftarrow \text{round}(\text{rand}(1, 2) \odot [H, W])$ ▷ $H \times W$ is map size
    if FEASIBLEPOINT(x,map) **then**
        $V \leftarrow [V;\ x]$
    end if
end while
Roadmap construction (adjacency list)
Initialize E as a list of $k+2$ empty neighbor arrays
for $i = 1$ **to** $k + 2$ **do**
    for $j = i + 1$ **to** $k + 2$ **do**
        if CHECKPATH(V_i, V_j, map) **then**
            $E[i] \leftarrow E[i] \cup \{j\};\ E[j] \leftarrow E[j] \cup \{i\}$
        end if
    end for
end for
A-star over PRM graph
$open \leftarrow \left[(1,\ 0,\ \text{HEURISTIC}(V_1, goal),\ \cdot,\ -1)\right]$
$open[1].f \leftarrow open[1].g + open[1].h$
$closed \leftarrow [\,]$; $pathFound \leftarrow$ **false**
while $|open| > 0$ **do**
    $n \leftarrow$ record in $open$ with minimum f
    remove n from $open$
    if $n.idx = 2$ **then** $pathFound \leftarrow$ **true**; **break**
    end if

Algorithm C.12 PRM-based path planning — Part II

for each $j \in E[n.idx]$ **do**
  if $j \notin \{c.idx \mid c \in closed\}$ **then**
    $g' \leftarrow n.g + \text{DISTANCE}(V_{n.idx}, V_j)$
    $h' \leftarrow \text{HEURISTIC}(V_j, goal)$
    $f' \leftarrow g' + h'$
    $add \leftarrow$ **true**
    if $j \in \{o.idx \mid o \in open\}$ **then**
      let o be the record in $open$ with $o.idx = j$
      if $o.f < f'$ **then** $add \leftarrow$ **false**
      else
        remove o from $open$; $add \leftarrow$ **true**
      end if
    end if
    if add **then**
      push $(j,\ g',\ h',\ f',\ |closed|)$ into $open$
    end if
  end if
end for
append n to $closed$
end while
if $\neg pathFound$ **then**
  return *error: no path found*
end if
Path reconstruction
$path \leftarrow [V_{n.idx}]$ ▷ start from goal record n
$p \leftarrow n.parent$
while $p > 0$ **do**
  $path \leftarrow [V_{closed[p].idx};\ path]$
  $p \leftarrow closed[p].parent$
end while
(Optional) Plot map, roadmap edges, and final path
return $path$

Algorithm C.13 Fuzzy Logic-based path planning

Input: BMP map file, source and goal coordinates, fuzzy inference system
Output: Path from source to goal avoiding obstacles
Load map from BMP file and convert to binary
Initialize robot parameters: size, speed, max speed, safety distance S, max acceleration, scaling factors
Load fuzzy inference system
Set current position $\leftarrow$ source, direction $\leftarrow$ initial heading
Initialize variables: $pathFound \leftarrow$ **false**, $prevTurn \leftarrow 0$, $pathCost \leftarrow 0$
while goal not reached **do**
    Measure distances to nearest obstacle in front, front-left, front-right
    Compute deviation angle to goal (normalized to $[-1, 1]$)
    Compute normalized distance to goal
    if distance to goal $<$ threshold **then** $pathFound \leftarrow$ **true**
    end if
    Determine preferred turn (rule-based pre-bias)
    Pass inputs $\{distanceFront, distanceFrontLeft, distanceFrontRight, angleGoal, turn, distanceGoal\}$ to FIS
    Compute steering adjustment and update robot orientation
    Compute safe distances (front and diagonals, with margin S)
    Derive max admissible speeds from acceleration and safe distances
    Update robot speed considering acceleration and limits
    if $robotSpeed = 0$ **then**
        return "Robot stopped to avoid collision"
    end if
    Update robot position based on current direction and speed
    if new position collides with obstacle **then**
        return "Collision recorded"
    end if
    Update path cost with traveled distance
    Plot robot and log current frame
end while
Return path, total cost, and processing time

Algorithm C.14 Learning-based path planning

Input: Dataset $D = \{X, y\}$, environment bounds, obstacles
Output: Best path from Start (x_s, y_s) to Goal (x_t, y_t)
Split dataset into training and testing sets
Choose K clusters (elbow method) ▷ only for clustering technique
Train clustering or regression model on $(X_{\text{train}}, y_{\text{train}})$
for each cluster k **do**
    Store mean damage $\mu_y[k]$
end for
for each obstacle o in environment **do**
    $features \leftarrow$ BUILDFEATURES(o)
    $predictedDamage \leftarrow$ PREDICT$(model, features)$
    if $predictedDamage >$ threshold **then**
        Mark o as *Avoid*
    end if
end for
function PATHCOST(path, model)
    $cost \leftarrow$ PATHLENGTH$(path)$
    for each collision c along $path$ **do**
        $damage \leftarrow$ PREDICT$(model,$ BUILDFEATURES$(c))$
        if $damage >$ threshold **then**
            re-plan path
        end if
    end for
    return $cost$
end function
Initialize swarm of particles with random waypoints
for iteration $= 1$ **to** MaxIter **do**
    for each particle p **do**
        $path \leftarrow$ COMPOSEPATH$(source, p.$waypoints$, goal)$
        $J \leftarrow$ PATHCOST$(path, model)$
        Update personal and global bests
        Update velocity and position ▷ algorithm rules
    end for
    Decay inertia weight w
end for
return global best path

Algorithm C.15 Multiple linear regression for damage estimation

Input: Dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, with $x_i \in \mathbb{R}^d$
Add bias term to form design matrix $X \in \mathbb{R}^{N\times(d+1)}$
Estimate parameters: $\hat{\beta} \leftarrow (X^\top X)^{-1} X^\top y$
function PREDICT(x_{new})
    $x_{\text{aug}} \leftarrow [1, x_{\text{new}}]$
    $\hat{y} \leftarrow x_{\text{aug}} \cdot \hat{\beta}$
    return $\hat{y}$
end function

Algorithm C.16 Clustering for damage estimation

Choose number of clusters K ▷ Elbow Method
Apply clustering to dataset ▷ K-Means
for each cluster k **do**
    Compute mean damage $\mu_y[k]$
end for
function PREDICT(x_{new})
    Find nearest cluster $k^\star$ for x_{new}
    $\hat{y} \leftarrow \mu_y[k^\star]$
    return $\hat{y}$
end function

Algorithm C.17 ANOVA for algorithm comparison

Collect timing results for each algorithm ▷ A-star, RRT, PRM, PSO
Create group labels for each data set
Perform one-way ANOVA: $(p, tbl, stats) \leftarrow$ ANOVA1 ▷ times, groups
Apply Tukey HSD test: $results \leftarrow$ MULTCOMPARE$(stats)$
Output comparison table: (Group1, Group2, MeanDiff, CI, pValue)

Bibliography

[1] Shubhani Aggarwal and Neeraj Kumar. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Comput. Commun.*, 149:270–299, 2020.

[2] Mary B. Alatise and G. Hancke. A review on challenges of autonomous mobile robot and sensor fusion methods. *IEEE Access*, 8:39830–39846, 2020.

[3] Adyan Nur Alfiyatin, Wayan Firdaus Mahmudy, and Yusuf Priyo Anggodo. K-means clustering and genetic algorithm to solve vehicle routing problem with time windows problem. *Indonesian Journal of Electrical Engineering and Computer Science*, 11:462–468, 8 2018.

[4] Alexander Amini, Igor Gilitschenski, Jacob Phillips, Julia Moseyko, Rohan Banerjee, S. Karaman, and D. Rus. Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *IEEE Robotics and Automation Letters*, 5:1143–1150, 2020.

[5] S. Aradi. Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23:740–759, 2020.

[6] R. Arvidson. Spirit mars rover mission to the columbia hills, gusev crater: Mission overview and selected results from the cumberland ridge to home plate. *Journal of Geophysical Research*, 113, 2008.

[7] Alala Bahamid, Azhar Mohd Ibrahim, Aisyah Ibrahim, Iman Zarifah Zahurin, and Azni Nabela Wahid. Intelligent robot-assisted evacuation: A review. *Journal of Physics: Conference Series*, 1706, 12 2020.

[8] Hollen Barmer, R. Dzombak, M. Gaston, Vijaykumar Palat, F. Redner, and Carol J. Smith. Human-centered ai. *IEEE Pervasive Comput.*, 22:7–8, 2021.

[9] Dario Bellicoso. Advances in real-world applications for legged robots. *Journal of Field Robotics*, 35:1311 – 1326, 2018.

[10] S. Bhattacharyya, D. Valeriani, C. Cinel, L. Citi, and R. Poli. Anytime collaborative brain–computer interfaces for enhancing perceptual group decision-making. *Scientific Reports*, 11, 2021.

[11] P. Bithas, E. T. Michailidis, Nikolaos Nomikos, D. Vouyioukas, and A. Kanatas. A survey on machine-learning techniques for uav-based communications. *Sensors (Basel, Switzerland)*, 19, 2019.

[12] M. Boban, M. Giordani, and M. Zorzi. Predictive quality of service: The next frontier for fully autonomous systems. *IEEE Network*, 35:104–110, 2021.

[13] Robert Bogue. Robots addressing agricultural labour shortages and environmental issues. *Ind. Robot*, 51:1–6, 2023.

[14] Joanna Bryson and Alan Winfield. Standardizing ethical design for artificial intelligence and autonomous systems. *Computer*, 50:116–119, 05 2017.

[15] Timo Bänziger, A. Kunz, and K. Wegener. Optimizing human–robot task allocation using a simulation tool based on standardized work descriptions. *Journal of Intelligent Manufacturing*, pages 1–14, 2020.

[16] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yuxin Pan, G. Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2019.

[17] Kuanqi Cai, Chaoqun Wang, Shuang Song, Haoyao Chen, and M. Meng. Risk-aware path planning under uncertainty in dynamic environments. *Journal of Intelligent and Robotic Systems*, 101, 2021.

[18] R. Calabrese and Belén Martín-Barragán. Interpretable machine learning for imbalanced credit scoring datasets. *Eur. J. Oper. Res.*, 312:357–372, 2023.

[19] Timothy K. Canham. The mars ingenuity helicopter - a victory for open-source software. *2022 IEEE Aerospace Conference (AERO)*, pages 01–11, 2022.

[20] Pei Cao, Yang Zhang, K. Zhou, and J. Tang. A reinforcement learning hyper-heuristic in multi-objective optimization with application to structural damage identification. *Structural and Multidisciplinary Optimization*, 66, 2022.

[21] Weiwen Cao, Xingyue Sun, Yajing Li, and Xu Chen. Multiaxial damage parameter evaluation by neural network-based symbolic regression. *Engineering Fracture Mechanics*, 327:112809, 2025.

[22] G. Ceccarelli, F. Catena, Pasquale Avella, Brian Wca Tian, F. Rondelli, Germano Guerra, Michele De Rosa, and Aldo Rocca. Emergency robotic surgery: the experience of a single center and review of the literature. *World Journal of Emergency Surgery : WJES*, 19, 2024.

[23] Suprava Chakraborty, Devaraj Elangovan, Padmasri Govindarajan, Mohamed F. Elnaggar, Mohammed M. Alrashed, and Salah Kamel. A comprehensive review of path planning for agricultural ground robots. *Sustainability*, 14(15), 2022.

[24] Rohan Chandra, Aniket Bera, and Dinesh Manocha. Using graph-theoretic machine learning to predict human driver behavior. *IEEE Transactions on Intelligent Transportation Systems*, 23:2572–2585, 2021.

[25] Shuo Chang, Yifan Zhang, Fan Zhang, Xiaotong Zhao, Sai Huang, Z. Feng, and Zhiqing Wei. Spatial attention fusion for obstacle detection using mmwave radar and vision sensor. *Sensors (Basel, Switzerland)*, 20, 2020.

[26] Long Chen, Yunxiao Shan, Wei Tian, Bijun Li, and Dongpu Cao. A fast and efficient double-tree rrt star-like sampling-based planner applying on mobile robotic systems. *IEEE/ASME Transactions on Mechatronics*, 23:2568–2578, 12 2018.

[27] Tianyi Chen, K. Zhang, G. Giannakis, and T. Başar. Communication-efficient policy gradient methods for distributed reinforcement learning. *IEEE Transactions on Control of Network Systems*, 9:917–929, 2022.

[28] Yutao Chen, Zhenyi Zhang, and Jie Huang. Dynamic task priority planning for null-space behavioral control of multi-agent systems. *IEEE Access*, 8:149643–149651, 2020.

[29] Yuxiao Chen, H. Peng, and J. Grizzle. Obstacle avoidance for low-speed autonomous vehicles with barrier function. *IEEE Transactions on Control Systems Technology*, 26:194–206, 2018.

[30] Yujiao Cheng, Liting Sun, and M. Tomizuka. Human-aware robot task planning based on a hierarchical task model. *IEEE Robotics and Automation Letters*, 6:1136–1143, 2021.

[31] F. Cherni, Y. Boutereaa, C. Rekik, and N. Derbel. Autonomous mobile robot navigation algorithm for planning collision-free path designed in dynamic environments. In *2015 JIEEEC 9th Jordanian International Electrical and Electronics Engineering Conference, JIEEEC 2015*, 2016.

[32] Sandeep P. Chinchali, Apoorva Sharma, James Harrison, Amine Elhafsi, Daniel Kang, Evgenya Pergament, Eyal Cidon, S. Katti, and M. Pavone. Network offloading policies for cloud robotics: a learning-based approach. *Autonomous Robots*, 45:997 – 1012, 2019.

[33] D. Claes and K. Tuyls. Multi robot collision avoidance in a shared workspace. *Autonomous Robots*, 42(8):1749–1770, 2018.

[34] C. Cockell. A low-diversity microbiota inhabits extreme terrestrial basaltic terrains and their fumaroles: Implications for the exploration of mars. *Astrobiology*, 19:284 – 299, 2019.

[35] Michele Colledanchise and L. Natale. On the implementation of behavior trees in robotics. *IEEE Robotics and Automation Letters*, 6:5929–5936, 2021.

[36] European Commission. European union artificial intelligence act (proposal). European Commission, 2021. Title III – High-Risk AI Systems.

[37] Jurek Czyzowicz, Kostantinos Georgiou, and Evangelos Kranakis. Group search and evacuation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11340 LNCS:335–370, 2019.

[38] Jeffrey Delmerico, Stefano Mintchev, Alessandro Giusti, Boris Gromov, Kamilo Melo, Tomislav Horvat, Cesar Cadena, Marco Hutter, Auke Ijspeert, Dario Floreano, Luca M. Gambardella, Roland Siegwart, and Davide Scaramuzza. The current state and future outlook of rescue robotics. *Journal of Field Robotics*, 36:1171–1191, 10 2019.

[39] Jory Denny, Read Sandström, Andrew Bregger, and Nancy M. Amato. Dynamic region-biased rapidly-exploring random trees. *Springer Proceedings in Advanced Robotics*, 13:640–655, 2020.

[40] Marianne Six Dijkstra, E. Siebrand, S. Dorrestijn, E. Salomons, M. Reneman, F. Oosterveld, R. Soer, D. Gross, and H. Bieleman. Ethical considerations of using machine learning for decision support in occupational health: An example involving periodic workers' health assessments. *Journal of Occupational Rehabilitation*, 30:343 – 353, 2020.

[41] Henghui Ding, Xudong Jiang, Bing Shuai, A. Liu, and Gang Wang. Semantic segmentation with context encoding and multi-path decoding. *IEEE Transactions on Image Processing*, 29:3520–3533, 2020.

[42] L. Ding, Shu Li, Yanjun Liu, H. Gao, Chaoyi Chen, and Z. Deng. Adaptive neural network-based tracking control for full-state constrained wheeled mobile robotic system. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47:2410–2419, 2017.

[43] K. Elbaz, S. Shen, W. Sun, Z. Yin, and A. Zhou. Prediction model of shield performance during tunneling via incorporating improved particle swarm optimization into anfis. *IEEE Access*, 8:39659–39671, 2020.

[44] V. Eshleman. Venus: Ionosphere and atmosphere as measured by dual-frequency radio occultation of mariner v. *Science*, 158:1678 – 1683, 1967.

[45] European Parliament and Council. Regulation (eu) 2024/... of the european parliament and of the council laying down harmonised rules on artificial intelligence (ai act). Official Journal of the European Union, 2024. Laying down harmonised rules on Artificial Intelligence (AI Act). Expected publication in the Official Journal shortly.

[46] N. Famaey, J. Vander Sloten, and E. Kuhl. A three-constituent damage model for arterial clamping in computer-assisted surgery. *Biomechanics and Modeling in Mechanobiology*, 12:123 – 136, 2012.

[47] Daniel Figurowski and Paweł Dworak. An algorithm for modeling an environment and creating a semantic model of its topology. *Procedia Computer Science*, 192:612–621, 2021.

[48] T. Fontainha, Leandro de Oliveira Silva, Wesley Moraes de Lira, A. Leiras, R. Bandeira, and L. F. Scavarda. Reference process model for disaster response operations. *International Journal of Logistics Research and Applications*, 25:1 – 26, 2020.

[49] G. Fragapane, R. Koster, F. Sgarbossa, and J. Strandhagen. Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *Eur. J. Oper. Res.*, 294:405–426, 2021.

[50] Yanzhou Fu, Austin Downey, L. Yuan, Tianyu Zhang, Avery Pratt, and Yunusa Balogun. Machine learning algorithms for defect detection in metal laser-based additive manufacturing: A review. *Journal of Manufacturing Processes*, 2022.

[51] Xinyu Gao, Jinhai Li, Lifeng Fan, Qiao Zhou, K. Yin, Jian-Xu Wang, Chao Song, Lan Huang, and Zhongyi Wang. Review of wheeled mobile robots' navigation problems and application prospects in agriculture. *IEEE Access*, 6:49248–49268, 2018.

[52] Esra Ersöz Genç and I. Genç. Enhancing disaster preparedness and response: Key strategies and interventions. *Disaster Medicine and Public Health Preparedness*, 19, 2025.

[53] Felipe L. Gewers, Gustavo R. Ferreira, H. F. D. Arruda, F. N. Silva, C. H. Comin, D. R. Amancio, and L. D. Costa. Principal component analysis. *ACM Computing Surveys (CSUR)*, 54:1 – 34, 2018.

[54] Zhijian Gou, Xiaozhao Jin, Bernt Johan Leira, Jin He, Dening Luo, Xiaona Xie, Tianpeng Huang, Jiekai Zou, and Jian Xie. Multi-target dynamic allocation and adaptive path planning for unmanned surface vehicles under marine surface with water flows. *Ocean Engineering*, 328:112330, 2025.

[55] P. Govender and V. Sivakumar. Application of k-means and hierarchical clustering techniques for analysis of air pollution: A review (1980–2019). *Atmospheric Pollution Research*, 11:40–56, 2020.

[56] Lucrezia Grassi, Mario Ciranni, Pierpaolo Baglietto, Carmine Tommaso Recchiuto, Massimo Maresca, and Antonio Sgorbissa. Emergency management through information crowdsourcing. *Information Processing and Management*, 60, 7 2023.

[57] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37:362–386, 4 2020.

[58] Sven Gronauer and K. Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55:895 – 943, 2021.

[59] Binghua Guo, Nan Guo, and Z. Cen. Obstacle avoidance with dynamic avoidance risk region for mobile robots in dynamic environments. *IEEE Robotics and Automation Letters*, 7:5850–5857, 2022.

[60] Shlomi Hacohen, Shraga Shoval, and Nir Shvalb. Probability navigation function for stochastic static environments. *International Journal of Control, Automation and Systems*, 17:2097–2113, 8 2019.

[61] Andreas H Hamel and Daniel Kostner. Multi-weight ranking for multi-criteria decision making. *Neural Computing and Applications*, 2023.

[62] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4:100–107, 1968.

[63] Allyson I. Hauptman, Beau G. Schelble, Nathan J. Mcneese, and K. Madathil. Adapt and overcome: Perceptions of adaptive autonomous agents for human-ai teaming. *Comput. Hum. Behav.*, 138:107451, 2022.

[64] A. Hentout, Abderraouf Maoudj, and M. Aouache. A review of the literature on fuzzy-logic approaches for collision-free path planning of manipulator robots. *Artificial Intelligence Review*, 56:3369–3444, 2022.

[65] H. Hewawasam, M. Y. Ibrahim, and G. Kahandawa. Past, present and future of path-planning algorithms for mobile robot navigation in dynamic environments. *IEEE Open Journal of the Industrial Electronics Society*, 3:353–365, 2022.

[66] D. Hobley. Formation of metre-scale bladed roughness on europa's surface by ablation of ice. *Nature Geoscience*, 11:901–904, 2018.

[67] Md. Naimul Hoque and Klaus Mueller. Outcome-explorer: A causality guided interactive visual interface for interpretable algorithmic decision making. *IEEE Transactions on Visualization and Computer Graphics*, 28:4728–4740, 2021.

[68] B. Horgan. The mineral diversity of jezero crater: Evidence for possible lacustrine carbonates on mars. *Icarus*, 2020.

[69] Seyedmohsen Hosseini and D. Ivanov. Bayesian networks for supply chain risk, resilience and ripple effect analysis: A literature review. *Expert Systems with Applications*, 161:113649 – 113649, 2020.

[70] Hao Ya Hsueh, Alexandru Iosif Toma, Hussein Ali Jaafar, Edward Stow, Riku Murai, Paul H.J. Kelly, and Sajad Saeedi. Systematic comparison of path planning algorithms using pathbench. *Advanced Robotics*, 36:566–581, 2022.

[71] Yingbai Hu, Guanglin Li, Peijiang Yuan, Chenguang Yang, and R. Song. Development of sensory-motor fusion-based manipulation and grasping control for a robotic hand-eye system. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47:1169–1180, 2017.

[72] B. Hynek. Implications for hydrologic processes on mars from extensive bedrock outcrops throughout terra meridiani. *Nature*, 431:156–159, 2004.

[73] IEEE. Ieee recommended practice for assessing the impact of autonomous and intelligent systems on human well-being. *IEEE Std 7010-2020*, pages 1–96, 2020.

[74] IEEE. Ieee standard model process for addressing ethical concerns during system design. *IEEE Std 7000-2021*, pages 1–82, 2021.

[75] Behnam Irani, Jingchuan Wang, and Weidong Chen. A localizability constraint-based path planning method for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 20:2593–2604, 2019.

[76] Rafiqul Islam, H. Habibullah, and Tagor Hossain. Agri-slam: a real-time stereo visual slam for agricultural environment. *Autonomous Robots*, 47:649 – 668, 2023.

[77] Robotics — safety requirements — part 1: Industrial robots, 2025.

[78] Robotics — safety requirements — part 2: Industrial robot applications and robot cells, 2025.

[79] Safety of machinery — general principles for design — risk assessment and risk reduction, 2010. Published November 2010; confirmed 2022; 77 pages.

[80] Security and resilience — emergency management — guidelines for incident management, 2018. Standard confirmed in 2024; 20 pages.

[81] Risk management — guidelines, 2018. Published February 2018; confirmed in 2023.

[82] Information technology — artificial intelligence — guidance on risk management, 2023. Edition 1, published 2023-02; ISO/IEC JTC 1/SC 42.

[83] Yuanyuan Jiang, Jie Zhang, Ling Xia, and Yanbin Liu. State of health estimation for lithium-ion battery using empirical degradation and error compensation models. *IEEE Access*, 8:123858–123868, 2020.

[84] Rico Jonschkowski and O. Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39:407–428, 2015.

[85] K. Kabassi and M. Virvou. Combining decision-making theories with a cognitive theory for intelligent help: A comparison. *IEEE Transactions on Human-Machine Systems*, 45:176–186, 2015.

[86] Júlia Kafková, Pavol Gažovič, V. Šimák, Pavol Kuchár, R. Pirník, and Silvia Di Girolamo. Enhancing emergency response: The leg-pulling robot for efficient evacuations. *2024 ELEKTRO (ELEKTRO)*, pages 1–6, 2024.

[87] Erlong Kang, H. Qiao, Jie Gao, and Wenjing Yang. Neural network-based model predictive tracking control of an uncertain robotic manipulator with input constraints. *ISA transactions*, 109:89–101, 2020.

[88] Mehmet Kara. A comprehensive framework for emergency in robotic environments. *IEEE Access*, 11:82539–82547, 2023.

[89] Mehmet Kara. Evaluation of popular path planning algorithms. *International Journal of Electronics and Telecommunications*, 70(1):85–92, 2024.

[90] Mehmet Kara. Planetary rovers on extreme terrains: Emergencies and responses. In *6th Space Resources Conference*. Springer Nature, 2024.

[91] Mehmet Kara. How planetary robots utilize machine learning for immediate decision. In *7th Space Resources Conference*. Springer Nature, 2025.

[92] S. Karimzadeh, A. Askan, M. Erberik, and A. Yakut. Seismic damage assessment based on regional synthetic ground motion dataset: a case study for erzincan, turkey. *Natural Hazards*, 92:1371–1397, 2018.

[93] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80:8091–8126, 2 2021.

[94] Kornél Katona, Husam A. Neamah, and Péter Korondi. Obstacle avoidance and path planning methods for autonomous navigation of mobile robot. *Sensors (Basel, Switzerland)*, 24, 2024.

[95] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[96] Mahdi Keykhaei, N. Samany, Mohammadreza Jelokhani-Niaraki, and S. Zlatanova. A situation-aware emergency evacuation (saee) model using multi-agent-based simulation for crisis management after earthquake warning. *Geo-spatial Information Science*, 27:1800 – 1823, 2023.

[97] S. Khaki, Lizhi Wang, and S. Archontoulis. A cnn-rnn framework for crop yield prediction. *Frontiers in Plant Science*, 10, 2019.

[98] Asif Khan, B. Rinner, and A. Cavallaro. Cooperative robots to observe moving targets: Review. *IEEE Transactions on Cybernetics*, 48:187–198, 2018.

[99] F. Khelladi, Mohamed Boudali, R. Orjuela, M. Cassaro, M. Basset, and C. Roos. An emergency hierarchical guidance control strategy for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23:4319–4330, 2022.

[100] Jong Hae Kim. Multicollinearity and misleading statistical results. *Korean Journal of Anesthesiology*, 72:558 – 569, 2019.

[101] B. R. Kiran, Ibrahim Sobh, V. Talpaert, P. Mannion, A. A. Sallab, S. Yogamani, and P. P'erez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23:4909–4926, 2020.

[102] Tobias Klamt, Malgorzata Kamedula, Hakan Karaoguz, Navvab Kashiri, Arturo Laurenzi, Christian Lenz, Daniele Leonardis, Enrico Mingo Hoffman, Luca Muratore, Dmytro Pavlichenko, Francesco Porcini, Diego Rodriguez, Zeyu Ren, Fabian Schilling, Max Schwarz, Massimiliano Solazzi, Michael Felsberg, Antonio Frisoli, Michael Gustmann, Patric Jensfelt, Klas Nordberg, Juergen Rossmann, Lorenzo Baccelliere, Uwe Suess, Nikos G. Tsagarakis, Sven Behnke, Xi Chen, Domenico Chiaradia, Torben Cichon, Massimiliano Gabardi, Paolo Guria, and Karl Holmquist. Flexible disaster response of tomorrow: Final presentation and evaluation of the centauro system. *IEEE Robotics and Automation Magazine*, 26:59–72, 12 2019.

[103] L. P. Knauth. Impact origin of sediments at the opportunity landing site on mars. *Nature*, 438:1123–1128, 2005.

[104] Y. Kondratenko, I. Atamanyuk, I. Sidenko, G. Kondratenko, and Stanislav Sichevskyi. Machine learning techniques for increasing efficiency of the robot's sensor and control information processing. *Sensors (Basel, Switzerland)*, 22, 2022.

[105] M. Korkmaz. A study over the general formula of regression sum of squares in multiple linear regression. *Numerical Methods for Partial Differential Equations*, 37:406 – 421, 2020.

[106] Mehmet Korkmaz and Akif Durdu. Comparison of optimal path planning algorithms. *14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, TCSET 2018 - Proceedings*, 2018-April:255–258, 4 2018.

[107] D. Koung, I. Fantoni, O. Kermorgant, and L. Belouaer. Consensus-based formation control and obstacle avoidance for nonholonomic multi-robot systems. In *16th IEEE International Conference on Control, Automation, Robotics and Vision, ICARCV 2020*, pages 92–97, 2020.

[108] Rafal Krzysiak and S. Butail. Information-based control of robots in search-and-rescue missions with human prior knowledge. *IEEE Transactions on Human-Machine Systems*, 52:52–63, 2022.

[109] B. Kuang, Chengzhen Gu, Z. Rana, Yifan Zhao, Shuang Sun, and Somtochukwu Godfrey Nnabuife. Semantic terrain segmentation in the navigation vision of planetary rovers—a systematic literature review. *Sensors (Basel, Switzerland)*, 22, 2022.

[110] Simge Küçükyavuz and Ruiwei Jiang. Chance-constrained optimization under limited distributional information: A review of reformulations based on sampling and distributional robustness. *EURO J. Comput. Optim.*, 10:100030, 2021.

[111] D. Lakens and Aaron R. Caldwell. Simulation-based power analysis for factorial analysis of variance designs. *Advances in Methods and Practices in Psychological Science*, 4, 2021.

[112] C. Lammie, A. Olsen, Tony Carrick, and Mostafa Rahimi Azghadi. Low-power and high-speed deep fpga inference engines for weed classification at the edge. *IEEE Access*, 7:51171–51184, 2019.

[113] S. Lebonnois. The deep atmosphere of venus and the possible role of density-driven separation of co2 and n2. *Nature Geoscience*, 10:473–477, 2017.

[114] D. Lee, Sang Su Lee, C. Ahn, Peng Shi, and C. Lim. Finite distribution estimation-based dynamic window approach to reliable obstacle avoidance of mobile robot. *IEEE Transactions on Industrial Electronics*, 68:9998–10006, 2021.

[115] Haoran Li, Qichao Zhang, and Dongbin Zhao. Deep reinforcement learning-based automatic exploration for navigation in unknown environment. *IEEE Transactions on Neural Networks and Learning Systems*, 31:2064–2076, 2020.

[116] Tianxu Li, Kun Zhu, Nguyen Cong Luong, Dusist Niyato, Qi hui Wu, Yang Zhang, and Bing Chen. Applications of multi-agent reinforcement learning in future internet: A comprehensive survey. *IEEE Communications Surveys and Tutorials*, 24:1240–1279, 2021.

[117] Xiaomei Li, Xiong Deng, Xiaoyong Wu, and zhijiang xie. Ss-detr: a strong sensing detr road obstacle detection model based on camera sensors for autonomous driving. *Measurement Science and Technology*, 36, 2025.

[118] Xinyu Li, Yuan He, and Xiaojun Jing. A survey of deep learning-based human activity recognition in radar. *Remote. Sens.*, 11:1068, 2019.

[119] Yonggang Li, Rencai Jin, Xiangrong Xu, Yuan yuan Qian, Haiyan Wang, Sha-Sha Xu, and Zhixiong Wang. A mobile robot path planning algorithm based on improved a* algorithm and dynamic window approach. *IEEE Access*, PP:1–1, 2022.

[120] Wei Liang and Haitao Xing. Investigations on speed planning algorithm and trajectory tracking control of intersection scenarios without traffic signs. *IEEE Access*, 11:8467–8480, 2023.

[121] Minas Liarokapis and A. Dollar. Combining analytical modeling and learning to simplify dexterous manipulation with adaptive robot hands. *IEEE Transactions on Automation Science and Engineering*, 16:1361–1372, 2019.

[122] Xi Lin, Yuan-Jin Yu, Shihan Chen, and Yang Shi. An improved apf method for complex and dynamic obstacles' avoidance. *Unmanned Syst.*, 11:175–189, 2022.

[123] Fan Liu and Yong Deng. Determine the number of unknown targets in open world based on elbow method. *IEEE Transactions on Fuzzy Systems*, 29:986–995, 2021.

[124] Lixing Liu, Xu Wang, Xin Yang, Hongjie Liu, Jianping Li, and Pengfei Wang. Path planning techniques for mobile robots: Review and prospect. *Expert Syst. Appl.*, 227:120254, 2023.

[125] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107:1697–1716, 2019.

[126] P. Lou, Kun Xu, Xuemei Jiang, Zheng Xiao, and Junwei Yan. Path planning in an unknown environment based on deep reinforcement learning with prior knowledge. *J. Intell. Fuzzy Syst.*, 41:5773–5789, 2021.

[127] Phan Gia Luan. Real-time hybrid navigation system-based path planning and obstacle avoidance for mobile robots. *Applied Sciences*, 2020.

[128] K. Lundeen, V. Kamat, C. Menassa, and W. McGee. Autonomous motion planning and task execution in geometrically adaptive robotized construction work. *Automation in Construction*, 2019.

[129] Wenqiang Luo, Qiming Liao, Siyuan Chen, and Jianhui Wu. Hyperspectral image classification with contrastive self-supervised learning under limited labeled samples. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2022.

[130] Pengfei Lv, Bingqing Wang, Feng Cheng, and Jinlin Xue. Multi-objective association detection of farmland obstacles based on information fusion of millimeter wave radar and camera. *Sensors (Basel, Switzerland)*, 23, 2022.

[131] Wenkai Lv, Pengfei Yang, Yunqing Ding, Zhenyi Wang, Chengmin Lin, and Quan Wang. Energy-efficient and qos-aware computation offloading in geo/leo hybrid satellite networks. *Remote. Sens.*, 15:3299, 2023.

[132] Yang Lv, Jinlong Lei, and Peng Yi. A local information aggregation-based multiagent reinforcement learning for robot swarm dynamic task allocation. *IEEE transactions on neural networks and learning systems*, 36(12):15888–15901, 2025.

[133] D. Macharet, A. A. Neto, V. Neto, and M. Campos. Nonholonomic path planning optimization for dubins' vehicles. *2011 IEEE International Conference on Robotics and Automation*, pages 4208–4213, 2011.

[134] R. Maidana, S. D. Kristensen, I. Utne, and A. J. Sørensen. Risk-based path planning for preventing collisions and groundings of maritime autonomous surface ships. *Ocean Engineering*, 279:114417, 2023.

[135] B. Malphrus. The lunar icecube em-1 mission: Prospecting the moon for water ice. *IEEE Aerospace and Electronic Systems Magazine*, 34:6–14, 2019.

[136] Bharadwaj R. K. Mantha, C. Menassa, and V. Kamat. Robotic data collection and simulation for evaluation of building retrofit performance. *Automation in Construction*, 2018.

[137] Huitan Mao and J. Xiao. Real-time conflict resolution of task-constrained manipulator motion in unforeseen dynamic environments. *IEEE Transactions on Robotics*, 35:1276–1283, 2019.

[138] Fei Meng, Liangliang Chen, Han Ma, Jiankun Wang, and M. Meng. Nr-rrt: Neural risk-aware near-optimal path planning in uncertain nonconvex environments. *IEEE Transactions on Automation Science and Engineering*, 21:135–146, 2022.

[139] Ahmad Merei, Hamid Mcheick, Alia Ghaddar, and D. Rebaine. A survey on obstacle detection and avoidance methods for uavs. *Drones*, 9(3):203, 2025.

[140] R. Milliken. Wind-blown sandstones cemented by sulfate and clay minerals in gale crater, mars. *Geophysical Research Letters*, 41:1149 – 1154, 2014.

[141] P. A. Mohan. Autonomous search and rescue robot. *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*, 2025.

[142] Anita Nanda, B. B. Mohapatra, A. P. Mahapatra, Abiresh Prasad Kumar Mahapatra, and A. Mahapatra. Multiple comparison test by tukey's honestly significant difference (hsd): Do the confident level control type i error. *International Journal of Statistics and Applied Mathematics*, 6(1):5–11, 2021.

[143] Mollik Nayyar and Alan R. Wagner. Effective robot evacuation strategies in emergencies. *2019 28th IEEE International Conference on Robot and Human Interactive Communication, RO-MAN 2019*, 10 2019.

[144] Milad Nazarahari, Esmaeel Khanmirza, and Samira Doostie. Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. *Expert Systems with Applications*, 115:106–120, 1 2019.

[145] Giang T. Nguyen, S. Dlugolinsky, Martin Bobak, V. Tran, A. García, Ignacio Heredia, Peter Malik, and L. Hluchy. Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey. *Artificial Intelligence Review*, 52:77–124, 2019.

[146] Shida Nie, Yujia Xie, Congshuai Guo, Hui Liu, Fawang Zhang, and Rui Liu. Personalized off-road path planning based on internal and external characteristics for obstacle avoidance. *IEEE Transactions on Intelligent Transportation Systems*, 26:2397–2409, 2025.

[147] Hanlin Niu, A. Savvaris, A. Tsourdos, and Ze Ji. Voronoi-visibility roadmap-based path planning algorithm for unmanned surface vehicles. *Journal of Navigation*, 72:850 – 874, 2019.

[148] OECD. Oecd principles on artificial intelligence. OECD, 2019.

[149] Luiz F. P. Oliveira, A. Moreira, and Manuel Silva. Advances in agriculture robotics: A state-of-the-art review and challenges ahead. *Robotics*, 10:52, 2021.

[150] W. Pedrycz, A. Kruglov, Yaroslav Plaksin, Arina Cheverda, Ayomide Bakare, G. Succi, Y. Bugayenko, and Mirko Farina. Prioritizing tasks in software development: A systematic literature review. *PLOS ONE*, 18, 2023.

[151] Sebastian Porębski. Evaluation of fuzzy membership functions for linguistic rule-based classifier focused on explainability, interpretability and reliability. *Expert Syst. Appl.*, 199:117116, 2022.

[152] Lijun Qiao, Xiao Luo, and Qingsheng Luo. An optimized probabilistic roadmap algorithm for path planning of mobile robots in complex environments with narrow channels. *Sensors (Basel, Switzerland)*, 22, 2022.

[153] Hongwei Qin, Shiliang Shao, Ting Wang, Xiaotian Yu, Yi Jiang, and Zonghan Cao. Review of autonomous path planning algorithms for mobile robots. *Drones*, 7(3):211, 2023.

[154] L. Qin, Yucheng Tang, Ujjaval Gupta, and Jian Zhu. A soft robot capable of 2d mobility and self-sensing for obstacle detection and avoidance. *Smart Materials and Structures*, 27, 2018.

[155] Anis Naema Atiyah Rafai, Noraziah Adzhar, Nor Izzati Jaini, et al. A review on path planning and obstacle avoidance algorithms for autonomous mobile robots. *Journal of Robotics*, 2022, 2022.

[156] N. Rajasekhar, T. K. Radhakrishnan, and N. Samsudeen. Exploring reinforcement learning in process control: a comprehensive survey. *International Journal of Systems Science*, 56(5):709–724, 2025.

[157] Amir Rasouli and John K. Tsotsos. Autonomous vehicles that interact with pedestrians: A survey of theory and practice. *IEEE Transactions on Intelligent Transportation Systems*, 21:900–918, 2018.

[158] Ankit A. Ravankar, Abhijeet Ravankar, Takanori Emaru, and Yukinori Kobayashi. Hpprm: Hybrid potential based probabilistic roadmap algorithm for improved dynamic path planning of mobile robots. *IEEE Access*, 8:221743–221766, 2020.

[159] Jian-Lin Ren, Jing Zhang, and Yani Cui. Autonomous obstacle avoidance algorithm for unmanned surface vehicles based on an improved velocity obstacle method. *ISPRS Int. J. Geo Inf.*, 10:618, 2021.

[160] Weibo Ren, Xiaonan Yang, Yan Yan, and Yaoguang Hu. The decision-making framework for assembly tasks planning in human–robot collaborated manufacturing system. *International Journal of Computer Integrated Manufacturing*, 36:289 – 307, 2022.

[161] Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adrià López Escoriza, R. J. Sloun, and Yonina C. Eldar. Kalmannet: Neural network aided kalman filtering for partially known dynamics. *IEEE Transactions on Signal Processing*, 70:1532–1547, 2021.

[162] D. Rose, J. Lyon, A. de Boon, Marc Hanheide, and S. Pearson. Responsible development of autonomous robotics in agriculture. *Nature Food*, 2:306 – 309, 2021.

[163] J. Ruiz-Sarmiento, C. Galindo, and Javier González Jiménez. Building multiversal semantic maps for mobile robot operation. *Knowl. Based Syst.*, 119:257–272, 2017.

[164] Daniela Rus and Michael T. Tolley. Design, fabrication and control of soft robots. *Nature*, 521:467–475, 5 2015.

[165] E. Salzano, M. Debernardi, Daniela Riccio, E. Danzi, A. Lolli, and L. Marmo. A case study of multiple explosions of chemicals under fire conditions. *Journal of Loss Prevention in the Process Industries*, 2019.

[166] Lorenzo Scalera, Andrea Giusti, R. Vidoni, and A. Gasparetto. Enhancing fluency and productivity in human-robot collaboration through online scaling of dynamic safety zones. *The International Journal of Advanced Manufacturing Technology*, 121:6783 – 6798, 2022.

[167] Simone Scardapane, Indro Spinelli, Atmesh Mahapatra, Abhinandan Singal, Divyansh Goel, Mufti Mahmud, Vikas Hassija, Kaizhu Huang, Amir Hussain, and V. Chamola. Interpreting black-box models: A review on explainable artificial intelligence. *Cognitive Computation*, 16:45–74, 2023.

[168] L. Schmid, Michael Pantic, R. Khanna, Lionel Ott, R. Siegwart, and Juan I. Nieto. An efficient sampling-based method for online informative path planning in unknown environments. *IEEE Robotics and Automation Letters*, 5:1500–1507, 2019.

[169] Max Schwarz, Tobias Rodehutskors, David Droeschel, Marius Beul, M. Schreiber, Nikita Araslanov, Ivan Ivanov, Christian Lenz, Jan Razlaw, Sebastian Schüller, David Schwarz, Angeliki Topalidou-Kyniazopoulou, and Sven Behnke. Nimbro rescue: Solving disaster-response tasks with the mobile manipulation robot momaro. *Journal of Field Robotics*, 34, 2017.

[170] A. Shakya, G. Pillai, and S. Chakrabarty. Reinforcement learning algorithms: A brief survey. *Expert Syst. Appl.*, 231:120495, 2023.

[171] Yanmei Shen and Bin Zhang. Advancing the emergency industry: Policy, innovation, and implications for national security. *Journal of the Knowledge Economy*, 15(4):5775–5799, 2024.

[172] Xiupeng Shi, Y. Wong, C. Chai, and Michael Zhi-Feng Li. An automated machine learning (automl) method of risk prediction for decision-making of autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 22:7145–7154, 2020.

[173] A. Skulimowski, Inez Badecka, Masoud Karimi, Pawel Lydek, and Przemysław Pukocz. Recent advances in artificial autonomous decision systems and their applications. In Marek Pawelczyk, Dariusz Bismor, Szymon Ogonowski, and Janusz Kacprzyk, editors, *Advanced, Contemporary Control*, pages 145–157, Cham, 2023. Springer Nature Switzerland.

[174] Andrzej Skulimowski. Predictive robotic harvesting based on non-stationary markov chain vegetation cycle model and crop simulation. *Proceedings of the Annual Hawaii International Conference on System Sciences*, pages 152–152(10), 2025.

[175] Andrzej Skulimowski and Victor Banuls. Ai alignment of disaster resilience management support systems. *Lecture Notes in Computer Science*, 12855:354–366, 10 2021.

[176] Andrzej M. J. Skulimowski. Anticipatory control of vehicle swarms with virtual supervision. In *Internet of Vehicles – Technologies and Services. IOV 2016*, volume 10036 of *Lecture Notes in Computer Science*, pages 65–81. Springer, Cham, 2016.

[177] Andrzej M. J. Skulimowski. Multicriteria decision planning with anticipatory networks to ensuring the sustainability of a digital knowledge platform. In Slim Ben Amor, Adiel Teixeira de Almeida, João L. de Miranda, and Emel Aktas, editors, *Advanced Studies in Multi-Criteria Decision Making*, chapter 9, pages 168–197. Chapman and Hall/CRC Press, Boca Raton, 2019.

[178] Andrzej M. J. Skulimowski. Anticipatory cooperation principles for autonomous space exploratory rovers. In Agata Kołodziejczyk, Joanna Pyrkosz-Pacyna, Krzysztof Grabowski, Katarzyna Malinowska, and Olga Sergijenko, editors, *Selected Proceedings of the 6th Space Resources Conference*, pages 171–195, Cham, 2024. Springer Nature Switzerland.

[179] Andrzej M. J. Skulimowski and Masoud Karimi. Intelligent anticipatory mobile robot networks for autonomous fruit harvesting. In A. Wojciechowski and P. Lipiński, editors, *Progress in Polish Artificial Intelligence Research 4*, number 2437 in Monografie Politechniki Łódzkiej, pages 121–140. Wydawnictwo Politechniki Łódzkiej, Łódź, Poland, 2023.

[180] Andrzej M.J. Skulimowski. Optimality and sensitivity of least-distance and avoidance solutions in multicriteria optimization. In *2018 23rd International Conference on Methods & Models in Automation & Robotics (MMAR)*, pages 375–380, 2018.

[181] Andrzej M.J. Skulimowski. Anticipatory networks: Concepts and selected applications. In *2025 25th International Conference on Control Systems and Computer Science (CSCS)*, pages 155–162, 2025.

[182] Andrzej M.J. Skulimowski, Inez Badecka, Arfa Hassan, Mehmet Kara, Paweł Łydek, and Przemysław Pukocz. Nowe metody analizy i wspomagania decyzji oraz ich zastosowania w inteligentnych systemach autonomicznych. In *AGH Monograph Series*. AGH Scientific Publishers, 2022.

[183] Andrzej MJ Skulimowski and Arkadiusz Ćwik. Communication quality in anticipatory vehicle swarms: A simulation-based model. In *Internet of Vehicles. Technologies and Services for Smart Cities: 4th International Conference, IOV 2017, Kanazawa, Japan, November 22-25, 2017, Proceedings 4*, pages 119–134. Springer, 2017.

[184] Andrzej M.J. Skulimowski, Przemysław Pukocz, Inez Badecka, and Mehmet Kara. A novel software architecture of anticipatory harvesting robot teams. In *25th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 47–52, 2021.

[185] Trevor Smith, Yuhao Chen, Nathan Hewitt, Boyi Hu, and Yu Gu. Socially aware robot obstacle avoidance considering human intention and preferences. *International Journal of Social Robotics*, 15:661 – 678, 2021.

[186] Qi Song, Qinglei Zhao, Shuxin Wang, Qiang Liu, and Xiaohe Chen. Dynamic path planning for unmanned vehicles based on fuzzy logic and improved ant colony optimization. *IEEE Access*, 8:62107–62115, 2020.

[187] Konduri Sriniketh, Anh Vu Le, Rajesh Elara Mohan, Bing J. Sheu, Vo Dinh Tung, Phan Van Duc, and Minh Bui Vu. Robot-aided human evacuation optimal path planning for fire drill in buildings. *Journal of Building Engineering*, 72, 8 2023.

[188] D. St-Onge, Marcel Kaufmann, Jacopo Panerati, Benjamin Ramtoula, Yanjun Cao, Emily B. J. Coffey, and G. Beltrame. Planetary exploration with robot teams: Implementing higher autonomy with swarm intelligence. *IEEE Robotics & Automation Magazine*, 27:159–168, 2020.

[189] S. Stahler. Seismic detection of the martian core. *Science*, 373:443 – 448, 2021.

[190] E. Steenstra. A possible high-temperature origin of the moon and its geochemical consequences. *Earth and Planetary Science Letters*, 538:116222, 2020.

[191] A. Stevens, S. Nawotniak, W. Garry, S. Payler, A. Brady, Matthew J. Miller, K. Beaton, C. Cockell, and D. Lim. Tactical scientific decision-making during crewed astrobiology mars missions. *Astrobiology*, 19:369 – 386, 2019.

[192] Changle Sun, Haitao Li, and Jun e Feng. Construction of interpretable hierarchical fuzzy systems subject to incomplete data. *Fuzzy Sets and Systems*, 2025.

[193] Yi Sun, Weitian Wang, Yi Chen, and Yunyi Jia. Learn how to assist humans through human teaching and robot learning in human–robot collaborative assembly. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52:728–738, 2022.

[194] Rafał Szczepański, T. Tarczewski, and Krystian Erwiński. Energy efficient local path planning algorithm based on predictive artificial potential field. *IEEE Access*, 10:39729–39742, 2022.

[195] J. Ricardo Sánchez-Ibáñez, Carlos J. Pérez-Del-pulgar, and Alfonso García-Cerezo. Path planning for autonomous mobile robots: A review. *Sensors*, 21, 12 2021.

[196] Zaid Tahir, Ahmed H. Qureshi, Yasar Ayaz, and Raheel Nawaz. Potentially guided bidirectionalized rrt* for fast optimal path planning in cluttered environments. *Robotics and Autonomous Systems*, 108:13–27, 10 2018.

[197] A. Tan, Han Yu, Li zhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE transactions on neural networks and learning systems*, PP, 2021.

[198] Gang Tang, Congqiang Tang, Christophe Claramunt, Xiong Hu, and Peipei Zhou. Geometric a-star algorithm: An improved a-star algorithm for agv path planning in a port environment. *IEEE Access*, 9:59196–59210, 2021.

[199] M. Tanveer, M. Sajid, Mushir Akhtar, A. Quadir, T. Goel, Aroof Aimen, Sushmita Mitra, Yu-Dong Zhang, Chin-Teng Lin, and J. Ser. Fuzzy deep learning for the diagnosis of alzheimer's disease: Approaches and challenges. *IEEE Transactions on Fuzzy Systems*, 32:5477–5492, 2024.

[200] T. Tasdizen, M. Hosseini, A. Umunnakwe, and M. Parvania. Intelligent damage classification and estimation in power distribution poles using unmanned aerial vehicles and convolutional neural networks. *IEEE Transactions on Smart Grid*, 11:3325–3333, 2020.

[201] V. Tereshchuk, John Stewart, Nikolay Bykov, Samuel Pedigo, S. Devasia, and A. Banerjee. An efficient scheduling algorithm for multi-robot task allocation in assembling aircraft structures. *IEEE Robotics and Automation Letters*, 4:3844–3851, 2019.

[202] Marco Tranzatto, Takahiro Miki, M. Dharmadhikari, Lukas Bernreiter, M. Kulkarni, Frank Mascarich, Olov Andersson, Shehryar Khattak, Marco Hutter, R. Siegwart, and K. Alexis. Cerberus in the darpa subterranean challenge. *Science Robotics*, 7, 2022.

[203] P. Trybała. Lidar-based simultaneous localization and mapping in an underground mine in złoty stok, poland. *IOP Conference Series: Earth and Environmental Science*, 942, 2021.

[204] O. Tsymbal, Artem Bronnikov, and A. Yerokhin. Adaptive decision-making for robotic tasks. *2019 IEEE 8th International Conference on Advanced Optoelectronics and Lasers (CAOL)*, pages 594–597, 2019.

[205] Zhigang Tu, Wei Xie, Dejun Zhang, R. Poppe, R. Veltkamp, Baoxin Li, and Junsong Yuan. A survey of variational and cnn-based optical flow techniques. *Signal Process. Image Commun.*, 72:9–24, 2019.

[206] Aizhan Tursunbayeva, C. Pagliari, Stefano Di Lauro, and Gilda Antonelli. The ethics of people analytics: risks, opportunities and recommendations. *Personnel Review*, 50(10):1348–1367, 2021.

[207] Michael Ehiedu Usiagwu, Mayowa Timothy Adesina, and Johnson Chinonso. Advanced machine learning models for real-time decision making in dynamic data environments. *International Journal of Science and Research Archive*, 14(2):101–112, 2025.

[208] Matías Vera, L. Vega, and P. Piantanida. Information flow in deep restricted boltzmann machines: An analysis of mutual information between inputs and outputs. *Neurocomputing*, 507:235–246, 2022.

[209] Shanu Verma, M. Pant, and V. Snasel. A comprehensive review on nsga-ii for multi-objective combinatorial optimization problems. *IEEE Access*, 9:57757–57791, 2021.

[210] V. Verma, M. Maimone, D. Gaines, R. Francis, T. Estlin, Stephen Kuhn, G. Rabideau, S. Chien, Michael M. McHenry, Evan Graser, A. Rankin, and Ellen Thiel. Autonomous robotics is driving perseverance rover's progress on mars. *Science Robotics*, 8, 2023.

[211] J. Villa, Jussi Taipalmaa, M. Gerasimenko, Alexander Pyattaev, Mikko Ukonaho, Honglei Zhang, Jenni Raitoharju, N. Passalis, Antti Perttula, Jussi Aaltonen, S. Andreev, Markus Aho, Sauli Virta, M. Gabbouj, M. Valkama, and K. Koskinen. acolor: Mechatronics, machine learning, and communications in an unmanned surface vehicle. *arXiv: Optimization and Control*, 2020.

[212] Caleb Wagner, Cecilia Mauceri, Philip Twu, Yuliya Marchetti, J. Russino, Dustin Aguilar, G. Rabideau, S. Tepsuporn, Steve Ankuo Chien, and Glenn Reeves. Demonstrating autonomy for complex space missions: A europa lander mission autonomy prototype. *Journal of Aerospace Information Systems*, 2023.

[213] Mohd Nadhir Ab Wahab, Amril Nazir, Ashraf Khalil, Wong Jun Ho, M. F. Akbar, Mohd Halim Mohd Noor, and Ahmad Sufril Azlan Mohamed. Improved genetic algorithm for mobile robot path planning in static environments. *Expert Syst. Appl.*, 249:123762, 2024.

[214] Jian Wang, Fan Li, Xuchong Zhang, and Hongbin Sun. Adversarial obstacle generation against lidar-based 3d object detection. *IEEE Transactions on Multimedia*, 26:2686–2699, 2024.

[215] Jian Wang, Chenqi Situ, and Mingzhu Yu. The post-disaster emergency planning problem with facility location and people/resource assignment. *Kybernetes*, 49:2385–2418, 2019.

[216] Xiaolu Wang, Xiaoping Zheng, and Yuan Cheng. Evacuation assistants: An extended model for determining effective locations and optimal numbers. *Physica A: Statistical Mechanics and its Applications*, 391:2245–2260, 3 2012.

[217] I. Wijegunawardana, S. Samarakoon, M. V. J. Muthugala, and M. R. Elara. Risk-aware complete coverage path planning using reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 55:2476–2488, 2025.

[218] Adam Williams, Bijo Sebastian, and Pinhas Ben-Tzvi. Review and analysis of search, extraction, evacuation, and medical field treatment robots. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 96:401–418, 12 2019.

[219] Florian Wirthmüller, Julian Schlechtriemen, Jochen Hipp, and M. Reichert. Teaching vehicles to anticipate: A systematic study on probabilistic behavior prediction using large data sets. *IEEE Transactions on Intelligent Transportation Systems*, 22:7129–7144, 2019.

[220] Zhengtian Wu, Jinyu Dai, Baoping Jiang, and H. Karimi. Robot path planning based on artificial potential field with deterministic annealing. *ISA transactions*, 2023.

[221] Aoran Xiao, Dayan Guan, Xiaoqin Zhang, and Shijian Lu. Domain adaptive lidar point cloud segmentation with 3d spatial consistency. *IEEE Transactions on Multimedia*, 26:5536–5547, 2024.

[222] Yuanlong Xie, Xiaolong Zhang, Wei Meng, Shiqi Zheng, Liquan Jiang, Jie Meng, and Shuting Wang. Coupled fractional-order sliding mode control and obstacle avoidance of a four-wheeled steerable mobile robot. *ISA transactions*, 2020.

[223] Yu xin Zhang, Jindong Wang, Yiqiang Chen, Hanchao Yu, and Tao Qin. Adaptive memory networks with self-supervised learning for unsupervised anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 35:12068–12080, 2022.

[224] Wenyuan Xu, Chen Yan, Weibin Jia, Xiaoyu Ji, and Jianhao Liu. Analyzing and enhancing the security of ultrasonic sensors for autonomous vehicles. *IEEE Internet of Things Journal*, 5:5015–5029, 2018.

[225] Zenglin Xu, Zixing Song, Xiangli Yang, and Irwin King. Graph-based semi-supervised learning: A comprehensive review. *IEEE Transactions on Neural Networks and Learning Systems*, 34:8174–8194, 2021.

[226] Qing Yang, Song Fu, Honggang Wang, and Hua Fang. Machine-learning-enabled cooperative perception for connected autonomous vehicles: Challenges and opportunities. *IEEE Network*, 35:96–101, 2021.

[227] Qinghua Yang, Xiaoqiang Du, Zhiheng Wang, Zhichao Meng, Zenghong Ma, and Qin Zhang. A review of core agricultural robot technologies for crop productions. *Comput. Electron. Agric.*, 206:107701, 2023.

[228] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11037–11045, 2020.

[229] E. Yanmaz, S. Yahyanejad, B. Rinner, H. Hellwagner, and C. Bettstetter. Drone networks: Communications, coordination, and sensing. *Ad Hoc Networks*, 68:1–15, 2018.

[230] Yuri D. V. Yasuda, Luiz Eduardo G. Martins, and F. Cappabianco. Autonomous visual navigation for mobile robots. *ACM Computing Surveys (CSUR)*, 53:1 – 34, 2020.

[231] Boris Yatsalo, Alexander Radaev, Elif Haktanir, Andrzej M.J. Skulimowski, and Cengiz Kahraman. A family of fuzzy multi-criteria sorting models ftopsis-sort: Features, case study analysis, and the statistics of distinctions. *Expert Systems with Applications*, 237:121486, 2024.

[232] M. Yazdi and T. Bouwmans. New trends on moving object detection in video images captured by a moving camera: A survey. *Comput. Sci. Rev.*, 28:157–177, 2018.

[233] Hao Ye, L. Liang, Geoffrey Y. Li, Joonbeom Kim, Lu Lu, and May Wu. Machine learning for vehicular networks: Recent advances and application examples. *IEEE Vehicular Technology Magazine*, 13:94–101, 2018.

[234] Nithin Sameer Yerramilli, N. Johnson, Omsri Sainadh Y Reddy, and S. Prajwal. Navigation systems using a*. *2021 International Conference on Recent Trends on Electronics, Information, Communication and Technology (RTEICT)*, pages 708–712, 2021.

[235] J. Yin, Ang Li, Tao Li, Wenxian Yu, and Danping Zou. M2dgr: A multi-sensor and multi-scenario slam dataset for ground robots. *IEEE Robotics and Automation Letters*, 7:2266–2273, 2021.

[236] Jinglun Yu. The path planning of mobile robot by neural networks and hierarchical reinforcement learning. *Frontiers in Neurorobotics*, 14, 2020.

[237] Angeliki Zacharaki, I. Kostavelis, A. Gasteratos, and I. Dokas. Safety bounds in human robot interaction: A survey. *Safety Science*, 127:104667, 2020.

[238] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965. Cited By :62358.

[239] Miguel A. Zamora-Izquierdo, José Santa, Juan A. Martínez, Vicente Martínez, and Antonio F. Skarmeta. Smart farming iot platform based on edge and cloud computing. *Biosystems Engineering*, 177:4–17, 1 2019.

[240] Fanyu Zeng, Chen Wang, and S. Ge. A survey on visual navigation for artificial agents with deep reinforcement learning. *IEEE Access*, 8:135426–135442, 2020.

[241] Hailun Zhang and Rui Fu. An ensemble learning–online semi-supervised approach for vehicle behavior recognition. *IEEE Transactions on Intelligent Transportation Systems*, 23:10610–10626, 2022.

[242] Hui Zhang, Yongfei Zhu, Xuefei Liu, and Xiangrong Xu. Analysis of obstacle avoidance strategy for dual-arm robot based on speed field with improved artificial potential field algorithm. *Electronics*, 2021.

[243] Qibin Zhang, Peng-Cheng Wang, and Zonghai Chen. An improved particle filter for mobile robot localization based on particle swarm optimization. *Expert Syst. Appl.*, 135:181–193, 2019.

[244] Tao Zhang and Quanyan Zhu. Dynamic differential privacy for admm-based distributed classification learning. *IEEE Transactions on Information Forensics and Security*, 12:172–187, 2017.

[245] Xuebo Zhang, Jiarui Wang, Yongchun Fang, and Jing Yuan. Multilevel humanlike motion planning for mobile robots in complex indoor environments. *IEEE Transactions on Automation Science and Engineering*, 16:1244–1258, 2019.

[246] Yan Zhang, Yu-Hao Wang, Xu-Hui Zhao, and Ruipeng Tong. Dynamic probabilistic risk assessment of emergency response for intelligent coal mining face system, case study: Gas overrun scenario. *Resources Policy*, 2023.

[247] Jianwei Zhao, Jianhua Fang, Shouzhong Wang, Kun Wang, Chengxiang Liu, and Tao Han. Obstacle avoidance of multi-sensor intelligent robot based on road sign detection. *Sensors (Basel, Switzerland)*, 21, 2021.

[248] Yecheng Zhao and Haibo Zeng. The concept of unschedulability core for optimizing priority assignment in real-time systems. *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 232–237, 2017.

[249] Huageng Zhong, Ming Cong, Minghao Wang, Yuqing Du, and Dong Liu. Hb-rrt:a path planning algorithm for mobile robots using halton sequence-based rapidly-exploring random tree. *Eng. Appl. Artif. Intell.*, 133:108362, 2024.

[250] Ziyuan Zhou, Guanjun Liu, and Ying-Si Tang. Multi-agent reinforcement learning: Methods, applications, visionary prospects, and challenges. *ArXiv*, abs/2305.10091, 2023.

[251] Yanming Zhu, Min Wang, Xuefei Yin, Jue Zhang, E. Meijering, and Jiankun Hu. Deep learning in diverse intelligent sensor based systems. *Sensors (Basel, Switzerland)*, 23, 2022.

[252] Baobao Zou, Chunxia Lu, Shirong Mao, and Yi Li. Effect of pedestrian judgement on evacuation efficiency considering hesitation. *Physica A: Statistical Mechanics and its Applications*, 547, 6 2020.

[253] Bipan Zou, Y. Gong, Xianhao Xu, and Zhe Yuan. Assignment rules in robotic mobile fulfilment systems for online retailers. *International Journal of Production Research*, 55:6175 – 6192, 2017.

[254] J. Zurn, Wolfram Burgard, and Abhinav Valada. Self-supervised visual terrain classification from unsupervised acoustic feature learning. *IEEE Transactions on Robotics*, 37:466–481, 2019.