Simulation-Based Testing and Runtime Monitoring for Autonomous Robotic Systems

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Sajad MazraehKhatiri

under the supervision of
Prof. Paolo Tonella and Dr. Sebastiano Panichella

January 2026

Dissertation Committee

Prof. Sebastian Elbaum	University of Virginia,Charlottesville, USA
Prof. Aitor Arrieta	Mondragon University, Mondragón, Spain
Prof. Carlo Alberto Furia	Università della Svizzera Italiana, Lugano, Switzerland
Prof. Piotr Krzysztof Didyk	Università della Svizzera Italiana, Lugano, Switzerland

Dissertation accepted on 27 January 2026

Research Advisor

Prof. Paolo Tonella

Co-Advisor

Dr. Sebastiano Panichella

PhD Program Director

Prof. Walter Binder & Prof. Stefan Wolf

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Sajad MazraehKhatiri
Winterthur, 27 January 2026

to the brave souls fighting for freedom in Iran, and to those we have lost in this enduring struggle.

to my parents, and to my beloved wife

“In theory, there is no difference
between theory and practice.

In practice, there is.”

—Attributed to
Yogi Berra & others

Abstract

The increasing deployment of autonomous robotic systems in critical applications necessitates robust methods for ensuring their safety and reliability. A primary obstacle to this is the *"reality gap"*, where simulation-based testing, a vital and scalable validation approach, often fails to represent real-world conditions, thus limiting its effectiveness in detecting critical failures before deployment. This dissertation addresses this challenge by proposing and validating a holistic framework for robotic system assurance, composed of two complementary layers: pre-deployment testing and runtime safety monitoring.

For pre-deployment assurance, this thesis introduces a novel search-based approach, *Surrealist*, which leverages operational data to first replicate real-world behaviors in simulation with high fidelity and then generate new challenging and realistic test scenarios that expose safety-critical failures. The effectiveness of this approach is first demonstrated on Unmanned Aerial Vehicles (UAVs), where it automatically discovers critical failure modes. The entire testing workflow is automated and orchestrated by *Aerialist*, a modular and scalable test bench developed as a core contribution of this dissertation.

The generalizability and practical value of the pre-deployment framework are then confirmed through a comprehensive industrial case study at ANYbotics. The framework was successfully adapted from UAVs to the ANYmal quadrupedal robot and integrated into their development workflow. It proved highly effective at uncovering algorithm deficiencies missed by manual methods and provided an objective, repeatable benchmark for comparing software versions, leading to its adoption as an essential pre-release validation gate. This initial success spurred the framework's further evolution into the company's MLOps pipeline, enabling rapid, large-scale benchmarking of new ML models by reducing test suite execution times from hours to minutes.

To complement pre-deployment testing, this dissertation introduces *Superialist*, a lightweight, black-box runtime monitor. Its design is grounded in a large-scale empirical study that established a quantifiable, moderate-to-strong correlation between observable *"decision uncertainty"* and subsequent safety violations, revealing that up to 89% of unsafe states are preceded by anomalous navigation patterns. Superialist

uses an autoencoder to detect these anomalous patterns in real-time with up to 96% precision, serving as an effective early warning system that identifies impending safety hazards up to 50 seconds in advance.

The contributions of this thesis provide a comprehensive, data-driven methodology to bridge the reality gap, enhancing robotic system safety through rigorous pre-deployment testing and real-time runtime monitoring. This work has also fostered broader community research through the establishment of the international UAV Testing Competition, which is built upon the frameworks developed in this dissertation.

Acknowledgements

This dissertation is the culmination of a long and challenging journey, one that would not have been possible without the guidance, support, and encouragement of many people. I would like to express my sincere gratitude to all of them.

First and foremost, I want to express my profound and sincere gratitude to my supervisors, Prof. Paolo Tonella and Dr. Sebastiano Panichella, who guided this dissertation from its inception. Paolo's insightful guidance, intellectual rigor, and unwavering support were the cornerstones of this work. He provided me with the freedom to explore my own ideas while always being available to steer me back on course with sharp questions and invaluable advice. His mentorship has shaped me as a researcher, and for that, I am truly thankful.

I owe an equal debt of gratitude to Sebastiano who has been more than a supervisor; he has been a true mentor, collaborator, and friend from day one. His infectious enthusiasm, boundless energy, and deep research expertise were a constant source of motivation. Thank you, Seba, for the countless hours of discussion, for pushing me to always do better, and for your steadfast belief in me.

I would like to extend my sincere thanks to the members of my dissertation committee: Prof. Sebastian Elbaum, Prof. Aitor Arrieta, Prof. Carlo Alberto Furia, and Prof. Piotr Krzysztof Didyk. I am honored to have had such a distinguished group of researchers evaluate my work. Your insightful feedback, challenging questions, and constructive criticism have been invaluable in refining this thesis.

My PhD journey was enriched by the colleagues I had the pleasure to work with. A special thanks to my colleague and friend, Christian Birchler. Our countless discussions, collaborations, and shared challenges were not only productive but also made the entire process much more enjoyable. I am also grateful to all my colleagues at the Test Automation Research Group (TAU) - Università della Svizzera Italiana, the Software Engineering Research Group (SERG) - University of Bern, and the Institute of Computer Science - Zurich of University of Applied Sciences (ZHAW), for creating a stimulating and supportive environment.

I also want to extend my sincere gratitude to my colleagues at ANYbotics. This thesis would not have been so impactful without your invaluable collaboration, openness,

and engagement. The opportunity to apply and evolve this research on a world-class industrial platform was a privilege. My special thanks go to Gabriel Hottiger, Maximilian Wulf, and the entire mobility and sensing team.

To all my friends, thank you. These five years away from home would not have been possible without your endless support, patience, and the much-needed escapes from the world of research. A special and heartfelt thanks to Mahdi and Maryam for always being there to share in all the significant moments, from the challenging lows to the celebratory highs.

My deepest gratitude goes to my family. To my wonderful parents, thank you for the countless sacrifices you made to raise me and provide the best possible environment for me to grow. You instilled in me the value of hard work and, most importantly, you made me believe that I could do this. To my brothers, thank you for your unconditional love and constant support throughout all these years. Your collective belief in me has been my foundation. Being so far from home has not always been easy, and I have missed you all dearly. I truly wish you could be here to celebrate this moment with me.

Finally, I want to express my most profound and heartfelt thanks to my lovely wife, Fatemeh. When I started this PhD, I thought I knew what the next five years would be about—long hours, difficult research, and finally, this document. Then I met you, and my entire life changed. while this dissertation is one of the greatest achievements of my life so far, finding love with you has been, without a doubt, the most significant and wonderful part of these five years. What makes our story so special is that we navigated this difficult path *together*, living the same shared life of challenges, frustrations, and triumphs. It was your immense patience, your unwavering support, and your endless love that became my greatest strength. Thank you for being my rock through all the ups and downs, for every sacrifice you made, and for every word of encouragement. I simply could not have done this without you.

Contents

Figures

Tables

Chapter 1

Introduction

Over the past decade, the proliferation of autonomous robotic systems has marked a transformative shift across numerous industrial and commercial sectors. Mobile robots, including Unmanned Aerial Vehicles (UAVs), legged robots, and autonomous ground vehicles, are no longer confined to research laboratories; they are actively being deployed for critical tasks such as automated logistics, large-scale infrastructure inspection, and hazardous search and rescue operations [36, 47]. As these systems become more autonomous and operate in complex, human-centric environments, ensuring their safety, reliability, and predictability has emerged as a paramount concern for researchers and practitioners alike [24, 33].

The conventional method for validating robotic systems–physical field trials–remains an indispensable step, but its significant drawbacks in cost, time, scale, and risk necessitate more efficient alternatives. Simulation-based testing has emerged as a powerful solution, offering a safe, low-cost, and scalable environment for automated and repeatable validation [3, 108]. However, its effectiveness is fundamentally constrained by a well-known challenge: the "reality gap" [41, 94]. This inherent discrepancy between a robot's behavior in a simulated environment and its performance in the physical world undermines the trustworthiness of simulation, making it difficult to generate truly representative test scenarios and reliably detect critical failures before deployment [4, 109]. Bridging this reality gap is the central research problem that this dissertation aims to overcome.

To address this, this thesis proposes and validates a holistic framework for robotic system assurance, illustrated in Figure 1.1. The framework is composed of two complementary layers: a comprehensive **Pre-deployment Safety Assurance** suite to enhance system robustness before deployment, and a **Runtime Safety Assurance** monitor to provide an essential layer of in-operation protection.

The Pre-deployment Safety Assurance layer is built upon two core components.

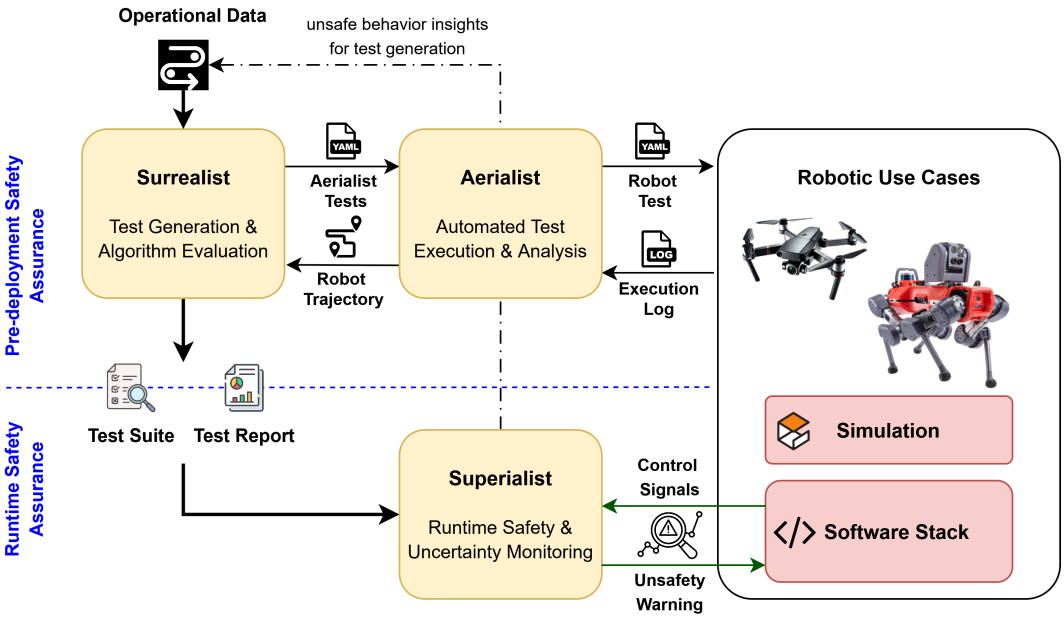

Figure 1.1. The holistic framework for robotic safety assurance presented in this dissertation, comprising a Pre-deployment Assurance suite (Aerialist and Surrealist) and a Runtime Assurance layer (Superialist).

The first is **Aerialist**, a modular and extensible test bench that automates the entire simulation-based testing workflow [59]. It provides a standardized interface for test definition and execution, enabling scalable and repeatable experimentation. The second component is **Surrealist**, a novel search-based test generation approach that uses real-world operational data to enhance the realism of simulation [58]. Surrealist first analyzes operational logs (including robot trajectory and sensor readings) to *replicate* real-world behaviors with high fidelity and then intelligently manipulates these scenarios to generate new, challenging test cases that expose safety-critical corner cases. Its effectiveness was first demonstrated on UAVs, where it consistently discovered previously unknown failure modes.

A cornerstone of this dissertation is the confirmation of the pre-deployment framework's generalizability and practical value through a comprehensive industrial case study at **ANYbotics**, a world leader in quadrupedal robotics. The framework was successfully adapted from UAVs to the ANYmal robot and integrated into the company's development workflow for a six-month evaluation [55]. During this period, the tools provided significant, tangible value to the engineering team. The automated test generation proved highly effective at uncovering critical algorithm deficiencies that had been missed by manual testing methods. The framework also provided an objective

and repeatable benchmark for quantitatively comparing the performance of different software versions, streamlining their development process. The usability and impact were so significant that the team adopted the framework as an essential pre-release validation gate, enhancing their overall verification pipeline and increasing their confidence in the system's robustness.

Finally, recognizing that pre-deployment testing alone is insufficient, the framework is complemented by a Runtime Safety Assurance layer embodied by **Superialist**, a lightweight, black-box runtime monitor [54]. Motivated by an empirical study establishing a link between observable "Decision Uncertainty" and unsafe states [54], Superialist detects anomalous control patterns in real-time. By training an autoencoder on nominal flight data, it achieves high precision (up to 96%) in detecting uncertainty and, more importantly, acts as an early warning system, predicting unsafe states up to 50 seconds in advance and providing a crucial window for corrective intervention. As illustrated in the framework architecture, the insights from runtime monitoring create a vital feedback loop: unsafe or uncertain scenarios identified by Superialist can be fed back into Surrealist. This enables a continuous improvement cycle where runtime failures are systematically reproduced and analyzed in simulation, and targeted regression tests are generated to help developers improve the system's robustness.

This dissertation makes the following primary contributions to the field of autonomous systems verification and validation:

- The design and implementation of a modular, general-purpose test bench for automating simulation-based testing of autonomous robotic systems.
- A novel search-based test generation approach that bridges the reality gap by creating realistic and challenging simulation scenarios from real-world data.
- A successful cross-domain adaptation and industrial validation of the testing framework, demonstrating its practical value, effectiveness, and impact in a leading industrial robotics environment.
- A lightweight, runtime safety supervisor that leverages decision uncertainty to accurately predict unsafe states in real-time.
- The establishment of a community-wide benchmark, the international UAV Testing Competition, built on this thesis's frameworks to advance test generation research.

The remainder of this dissertation is organized as follows. Chapter 2 reviews the state of the art in robotic systems testing. Chapter 3 introduces Aerialist, the foundational test automation framework, and the robotic use cases studied in this thesis.

Chapter 4 presents the Surrealist approach for realistic test generation, with a case study on UAVs, and discusses the UAV Testing Competition. Chapter 5 details the industrial adoption and cross-domain validation of the framework with quadrupedal robots at ANYbotics. Chapter 6 presents the Superialist runtime safety monitor, also validated on UAVs. Finally, Chapter 7 concludes the dissertation, summarizing the contributions and outlining directions for future research.

Main Publications The work presented in this thesis is based on research that has been published in the following peer-reviewed conference proceedings and journals:

1. Khatiri, S., Panichella, S., and Tonella, P. "Simulation-based testing of unmanned aerial vehicles with aerialist." In Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24), 2024.

2. Khatiri, S., Panichella, S., and Tonella, P. "Simulation-based test case generation for unmanned aerial vehicles in the neighborhood of real flights." In 2023 IEEE 16th International Conference on Software Testing, Verification and Validation (ICST), 2023.

3. Khatiri, S., et al. "Bridging Research and Practice in Simulation-based Testing of Industrial Robot Navigation Systems." 2025 IEEE/ACM International Conference on Automated Software Engineering (ASE), 2025.

4. Khatiri, S., et al. "When Uncertainty Leads to Unsafety: Empirical Insights into the Role of Uncertainty in Unmanned Aerial Vehicle Safety." Empirical Software Engineering (EMSE) Journal, 2025.

5. Khatiri, Sajad, et al. "ICST Tool Competition 2025–UAV Testing Track." 2025 IEEE Conference on Software Testing, Verification and Validation (ICST), 2025.

Chapter 2

State of the Art

The verification and validation of autonomous robotic systems is a complex and rapidly evolving field. As these systems are increasingly deployed in safety-critical applications, the demand for rigorous and scalable testing methodologies has intensified. This chapter reviews the state of the art in testing autonomous systems, establishing the context and motivation for the contributions of this dissertation. We first survey the fundamental challenges inherent in testing robotic systems, then delve into the specific problem of the "reality gap" in simulation-based testing. Subsequently, we examine existing approaches for automated test generation and conclude with a review of methods for runtime safety assurance.

2.1 Challenges in Testing Robotic Systems

Ensuring the proper behavior of autonomous robotic systems remains a significant open research challenge [4, 8]. Unlike traditional software, these systems are characterized by a continuous and complex interaction with a dynamic, unpredictable environment. This inherent complexity makes it difficult to define expected behaviors, create representative test oracles [68], and achieve adequate test coverage for certification processes [20, 43]. The continuous collection and analysis of real-time data to make runtime decisions [68, 108] further complicates the verification and validation process, making it difficult to determine whether a system is behaving as *expected* under all possible conditions.

Several empirical studies have highlighted the unique difficulties in this domain. A foundational study by Afzal, Le Goues, Hilton and Timperley [4] identified the *engineering complexity* of the test environment as a primary obstacle, emphasizing the difficulty of designing realistic inputs to thoroughly test the system. This finding is strongly echoed in a subsequent survey of robotics practitioners conducted by Afzal

et al. [3], who report that two of their top concerns are the *reality gap* between simulated and real-world behavior and the difficulty of *reproducing* failures encountered in the field within a controlled simulation environment.

These challenges are further substantiated by in-depth analyses of software bugs in real-world robotic systems. Wang et al. [109] studied bugs from popular UAV Autopilot platforms like PX4 and Ardupilot, creating a detailed taxonomy of their root causes. Their findings revealed that while developers heavily rely on simulators for reproducing and debugging issues, setting up simulation environments that are realistic enough to capture the same bugs observed in physical tests is both hard and expensive. Similarly, an empirical study by Timperley et al. [108] on fixed bugs in the Ardupilot [9] platform concluded that many bugs could theoretically be detected before field deployment if more effective simulation-based testing practices were in place.

Collectively, this body of research underscores a critical and unmet need in the field: the development of advanced methodologies and tools that can systematically manage the complexity of the test environment, facilitate the creation of realistic and challenging test scenarios, and ultimately bridge the persistent gap between simulation and reality.

2.2 Simulation-Based Testing and the Reality Gap

Simulation-based testing has emerged as the most promising approach to address the cost, safety, and scalability limitations of physical field testing [3, 108, 109]. However, its effectiveness is fundamentally limited by the *reality gap* [3, 4, 85, 109]. This discrepancy between a system's behavior in simulation and its performance in the physical world is a central challenge in robotics. Simulations are necessarily simplified for computational feasibility, yet must reflect real-world phenomena at a given level of veracity. This gap arises from the difficulty of accurately modeling complex physical interactions, such as the torque characteristics of actuators, sensor noise and latency, and the nuances of rendered images like reflections and textures [21].

The reality gap is a particularly acute problem in modern robotics, where practitioners increasingly rely on simulation to evaluate designs and train controllers using techniques like reinforcement learning [41, 94]. The task of transferring robot skills acquired in a simulated environment to a physical setting, a process widely referred to as *Sim2Real transfer*, remains a significant open challenge [25].

The research community has approached this problem from several angles, primarily focusing on the development phase of robotic controllers. *Domain Randomization* is a widely used technique that attempts to bridge the gap by exposing the control algorithm to a wide variety of randomized simulation environments, rather than a single,

fixed one. The underlying assumption is that if the controller is robust to a sufficiently diverse range of simulated conditions, the real world will appear as just another variation it can handle [19, 49, 67, 82]. Other approaches argue that pure simulation is insufficient and propose combining it with a small amount of real-world data to ground the simulation [18, 119]. This is often achieved by recalculating the fitness of select solutions in the real world and incorporating the observed deviation into the optimization process [65, 94]. Some methods also focus on tuning the parameters of the physics simulator itself to better match real-world observations for a specific scenario [21].

While these studies provide valuable solutions for addressing the reality gap during the *development* of robotic systems, far fewer have focused on its implications for the *testing* phase. A notable exception is the "world-in-the-loop" simulation proposed by Hildebrandt and Elbaum [39], which creates a mixed-reality environment by blending sensor data from both simulated and real-world sources to feed into the system under test. While powerful, this approach can be complex to set up and may not be suitable for all testing scenarios. The limitations of current techniques highlight the need for a testing methodology that can leverage the benefits of simulation while explicitly managing and mitigating the effects of the reality gap by generating test scenarios that are both realistic and challenging.

2.3 Automated Test Generation for Autonomous Systems

To improve test coverage and efficiency, particularly for complex systems controlled by Deep Learning (DL), researchers have adapted traditional testing techniques to address challenges like test input generation [22, 26, 34, 73, 74, 88, 107, 112], test oracle definition [48, 88, 103], and test adequacy [63, 74, 88]. A significant portion of this work focuses on generating diverse and effective test scenarios in simulation. These approaches can be broadly distinguished by their strategy for exploring the vast space of possible test environments: undirected versus directed generation.

Undirected methods, such as procedural content generation and fuzzing, aim to create a wide variety of test scenarios through randomization. Procedural content generation, as explored by Sotiropoulos et al. [101], uses randomization to create open-space worlds to find navigation bugs, with a later focus on assessing the difficulty of these worlds [100]. To provide more structure, Parra et al. [87] introduced FloorPlan DSL for defining indoor test environments and Variation DSL for structured variability, such as sampling obstacle sizes from a normal distribution. Fuzzing methods, most notably PHYS-FUZZ by Woodlief et al. [111], systematically vary environmental parameters and robot poses to uncover crashes in mobile robots. Similarly, tools like *Local Planner Bench* [102] support randomized environments to benchmark local ob-

stacle avoidance algorithms. While these methods excel at creating a wide variety of test scenarios for broad coverage, they are generally undirected and can be inefficient at finding the specific, complex failure modes that arise from intricate interactions between the robot and its environment.

In contrast, directed methods, particularly those using search-based techniques, offer a more guided approach to failure discovery. These approaches formulate test generation as an optimization problem, using a fitness function to guide the search towards more challenging or failure-inducing scenarios. This strategy is prominent in the autonomous driving domain, where a rich body of work exists on generating critical test cases. Researchers such as Gambi et al. [30], Stocco et al. [103], Riccio and Tonella [92], Birchler et al. [15], Birchler et al. [16], and Abdessalem et al. [1] have successfully employed search algorithms to intelligently explore the vast parameter space of driving environments—including road layouts, traffic behavior, and sensor conditions—to find configurations that violate safety requirements. The success of this paradigm is leading to its adoption in other robotic domains. For instance, MARTENS, proposed by Humeniuk et al. [44], is a search-based method for testing the DL vision models used in autonomous manipulators. This body of work demonstrates the power and efficiency of guided search in systematically and purposefully evolving scenarios to find critical failures in complex autonomous systems.

2.4 Runtime Monitoring and Safety Assurance

Recognizing that exhaustive pre-deployment testing is infeasible for autonomous systems, runtime monitoring has become a critical layer of *safety assurance* [38]. The goal is to detect anomalous or unsafe behavior during operation, enabling corrective actions to be taken before a failure materializes, forming the basis for self-healing or adaptive control systems that can re-plan or enter a safe mode when an anomaly is detected [12, 75, 113]. This concept is formalized in the emerging field of *Dynamic Safety Assurance (DSA)*, which advocates for the continuous, through-life management of a system's safety case [23, 93]. Unlike traditional, static safety cases developed at design time, DSA frameworks integrate runtime evidence to continuously assess risks and the validity of safety arguments, with runtime monitors being a cornerstone of this process.

Research into runtime monitoring for robotic systems has largely followed two parallel threads: anomaly detection and uncertainty quantification. **Anomaly detection** approaches aim to identify deviations from a learned model of normal behavior. Early work in this area focused on detecting anomalies in individual components, such as sensors [29], motors [71], or communication networks [5, 37, 115]. More recent and

sophisticated approaches have shifted to system-level monitoring. For instance, Sindhwani et al. [97] developed an anomaly detection framework for delivery drones that learns normal mission behavior from a combination of sensor and control signals. Similarly, Shar et al. [95] used an LSTM-based model to detect sequences of flight states that could lead to physical instability. A common characteristic of these approaches is their reliance on data from internal sensors and actuators to build a model of the system's nominal state.

A related and more powerful concept is **uncertainty quantification**, which aims to measure a system's confidence in its own perceptions and decisions. This has been extensively studied for Deep Learning (DL) components, where techniques like Bayesian Neural Networks can provide a rigorous framework for modeling uncertainty [10, 79]. However, a key challenge is that the internal uncertainty of a perception model (e.g., in computer vision) is often not propagated through the decision-making pipeline to the final control output [10]. To circumvent this, black-box methods have been proposed that infer uncertainty from the system's inputs or outputs, without needing access to its internal state. A seminal example is the work of Stocco et al. [103], who used an autoencoder to monitor the input camera feed of a simulated self-driving car. By learning to reconstruct images from a "normal" driving context, their system could detect novel or unexpected scenes by measuring the reconstruction error, thereby predicting potential misbehaviors. While these studies show the promise of data-driven, black-box techniques for runtime monitoring, the existing literature has yet to fully explore the relationship between observable uncertainty in a robot's high-level *control signals* and its subsequent safety state.

Chapter 3

Test Automation for Autonomous Robotic Systems

The preceding chapter established the significant challenges in verifying and validating autonomous robotic systems, from the complexity of their interaction with the physical world to the persistent reality gap in simulation. Addressing these challenges requires a foundational infrastructure that can automate the tedious and complex aspects of testing, allowing researchers and developers to focus on higher-level concerns like test case design and system behavior analysis.

This chapter introduces *Aerialist* [59], the foundational framework developed as part of this dissertation to provide such an infrastructure. While existing simulation-based testing frameworks focus on specific platforms or require manual orchestration, Aerialist provides a unified, scalable solution that abstracts low-level complexities, enabling systematic and reproducible testing across diverse robotic domains. We begin by detailing the core verification challenges rooted in the concept of robotic mobility. We then ground these challenges by presenting the two distinct and complex robotic platforms that serve as the primary use cases throughout this thesis: Unmanned Aerial Vehicles (UAVs) and industrial quadrupedal robots. Finally, we present the architecture and core capabilities of Aerialist, a general-purpose, extensible framework designed to automate the entire lifecycle of simulation-based testing for these and other autonomous robotic systems.

3.1 The Challenge of Verifying Robotic Mobility

Robotic mobility—the ability of a system to move purposefully and safely through its environment—is the core enabling capability for autonomous systems in their most complex and safety-critical applications. The expected impact of this technology is vast,

with applications ranging from search and rescue robots navigating unstructured disaster sites, to rovers performing space exploration on distant planets, drones conducting large-scale crop monitoring, and quadrupedal robots inspecting hazardous industrial facilities [31, 36, 98].

In all of these high-stakes contexts, the system's ability to reliably perceive its surroundings and navigate robustly is not just a feature; it is the fundamental prerequisite for mission success and safety. A failure in mobility—whether a collision, a fall, or an inability to find a path—is a catastrophic failure of the entire system. As autonomous systems become more integrated into these critical domains, ensuring the robustness of their mobility functions has therefore emerged as a primary verification and validation challenge [47]. **In this thesis, we focus specifically on the challenges of verifying robotic mobility,** treating it as a core pillar of system safety and reliability.

This capability is typically addressed through a functional hierarchy composed of two distinct but tightly coupled layers: *locomotion* and *navigation* [40]. **Locomotion** represents the low-level control of actuators to achieve stable and efficient movement. The strategies for locomotion vary dramatically with a robot's physical design and intended operational domain [35]. Wheeled and tracked robots, for example, excel on relatively smooth surfaces but are limited by rough terrain [98]. In contrast, aerial robots must manage complex 3D dynamics, while legged robots, such as quadrupeds, must execute diverse gaits like walking and trotting to navigate challenging environments like stairs and rubble [40, 81, 116]. Achieving this relies on advanced control methods, including Model Predictive Control (MPC) [70] and reinforcement learning [40, 114, 116], to ensure movements are reliable and stable across a wide range of conditions.

Complementary to this, **navigation** involves the higher-level, intelligent decision-making required for the robot to reach a destination safely and efficiently [35]. This process begins with perception, where sensor data from cameras, LiDAR, and IMUs are fused to build a model of the surrounding world [86]. Based on this model, the navigation system must plan optimal or near-optimal paths, avoid both static and dynamic obstacles, and adapt to unforeseen changes [86]. This planning is often divided into two stages: *global planning*, which maps a high-level route to the goal, and *local planning*, which manages real-time reactions and obstacle avoidance [86]. For many platforms, particularly legged robots, navigation and locomotion are intricately linked, as the choice of gait and dynamic capabilities directly influences how the robot can traverse terrain and avoid obstacles [27].

The intricate interplay between perception, planning, and control makes the verification of robotic mobility a formidable task. A failure in any part of this hierarchy can lead to mission failure or critical safety violations. Consequently, any effective testing framework must be capable of orchestrating complex test scenarios that systematically

stress all layers of the mobility stack, from low-level control stability to high-level navigation logic.

3.2 Robotic Platforms Under Test

To validate the generality and scalability of the proposed testing framework, this dissertation considers two representative and highly complex autonomous platforms: an Unmanned Aerial Vehicle (UAV) implementing a PX4-based flight control stack, and the ANYmal quadrupedal robot, designed for autonomous industrial inspection. These two types of systems—aerial robots and legged robots—were specifically chosen as they are among the most promising and widely adopted platforms for a variety of real-world applications, such as autonomous inspection, monitoring, and logistics.

While these platforms embody fundamentally different forms of mobility (aerial flight versus legged locomotion), this thesis focuses on a common and critical high-level challenge they both share: **waypoint-based navigation**. In both use cases, the core task for the robot is to autonomously plan and execute a safe path to a predefined navigation goal (or a sequence of waypoints), successfully maneuvering around any obstacles it encounters along the way. This specific pairing—testing an identical high-level navigation task on two platforms with completely different underlying physics—makes them the ideal testbed. Their inclusion serves to prove that the developed testing methodology is modular, successfully abstracts platform-specific details, and can handle diverse robotic architectures, simulation interfaces, and operational challenges.

3.2.1 Unmanned Aerial Vehicles (UAVs)

The first use case focuses on autonomous quadcopter UAVs, a domain characterized by complex flight dynamics and the critical need for reliable navigation in three-dimensional space. The experiments are built on the **PX4 Autopilot** [80], a popular open-source flight control platform widely adopted in both research and industry, running on a development kit like the one shown in Figure 3.1. PX4 provides a modular architecture that supports a range of flight modes, from manual assistance to high-level automation such as pre-planned waypoint missions. The primary investigated challenge is **autonomous obstacle avoidance**, which stresses the capabilities of the PX4-Avoidance module [91], a component that uses computer vision to perceive and navigate through cluttered environments.

The validation leverages PX4's **Software-in-the-Loop (SIL)** simulation capabilities, which enable the exact firmware flown on a physical drone to be executed and

Figure 3.1. The PX4 Vision Development Kit (v1.5), the quadcopter used in the UAV experiments, and its components [90].

tested within a safe, virtual environment. This is achieved by interfacing PX4 with a physics simulator that provides sensor data and receives actuator commands. Because the communication interface is identical for both real and simulated UAVs, the framework developed in this thesis can transparently automate test campaigns in either domain. It supports multiple PX4-compatible simulators, including **Gazebo** [64] and **jMAVSim**[90], ensuring flexibility. During execution, all flight data—including sensor readings, state estimations, and control commands—are recorded in detailed **flight logs**, which are automatically collected and parsed for post-flight analysis and failure diagnosis.

3.2.2 Industrial Quadrupedal Robots

The second use case extends the testing challenge to the domain of legged ground robots, which present a different set of complexities related to locomotion, terrain interaction, and navigation in confined industrial spaces. This study employs the **ANYmal D** [46] (Figure 3.2), a state-of-the-art quadrupedal robot developed by ANYbotics[1] for autonomous industrial inspection tasks. Its legged design enables mobility over human-centric terrains like stairs, which are inaccessible to wheeled or tracked robots. The ANYmal D is equipped with a sophisticated sensor suite, including a 360° LiDAR for mapping, wide-angle cameras for navigation, and multiple depth cameras for com-

[1] https://www.anybotics.com/

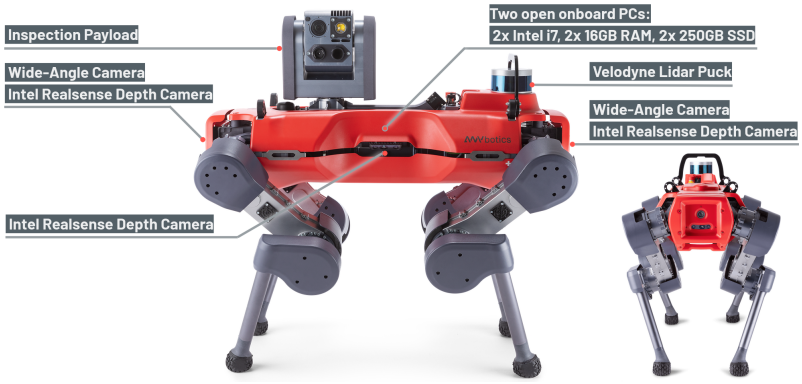

Figure 3.2. The ANYmal D quadrupedal robot, highlighting its sensor architecture and key components [7].

prehensive terrain perception. Its software stack is built upon the Robot Operating System (ROS).

The primary focus here is on **robust local planning and obstacle avoidance in cluttered environments**. The robot must safely navigate narrow corridors, doorways, and other obstacles typical of industrial facilities, a task requiring tight coordination between its perception, navigation, and locomotion systems. This use case leverages high-fidelity simulators, including **Gazebo** and **Isaac Gym** [76], with a custom, proprietary model of the ANYmal robot and its navigation stack as provided by ANYbotics. This setup allows for realistic, large-scale testing of the robot's complete software system in a controlled, virtual environment prior to physical deployment.

3.3 Aerialist: A General-Purpose Test Automation Framework

To address the significant engineering complexity involved in testing the robotic systems described above, this dissertation introduces **Aerialist**[2] [59], a novel, general-purpose framework for test automation. While initially developed for UAVs, Aerialist was designed from the ground up with a modular and extensible philosophy, allowing it to be adapted to other robotic platforms, as demonstrated by its successful application to the ANYmal quadruped. The core design philosophy of Aerialist is to **abstract away the low-level complexities of simulation-based testing**, providing a unified interface that automates the entire validation workflow. This allows developers and researchers to focus on the high-level logic of *what* to test, rather than the low-level implementation

[2] https://github.com/skhatiri/Aerialist

of *how* to run the test.

3.3.1 The Standardized Test Description Model

A fundamental challenge in robotics verification is that manual, system-level tests are often defined by a combination of implicit configurations, specific physical environment setups, and sequences of runtime commands. This makes them difficult to reproduce, version, and scale. To overcome this, Aerialist introduces a standardized, file-based **Test Description Model**. This model formalizes all necessary components of a test into a structured, human-readable YAML format, making test cases explicit, version-controllable, and portable across different execution environments. The model is composed of the following core properties:

- **Robot Configuration:** Specifies the hardware and software setup of the robot under test, including all necessary parameter values and firmware configurations.
- **Environment Configuration:** Defines the simulated world, including the choice of simulator, the simulation scene, weather conditions (e.g., wind), the initial position of the robot, and the properties of any objects in the scene, such as the size, position, and orientation of obstacles.
- **Mission Configuration:** The specific task of the robot during the test. It may include a predefined autonomous mission plan, or timestamped sequence of external commands to be sent to the robot at runtime, allowing for the simulation of operator inputs.
- **Test Oracle (Optional):** Defines the expected outcome or success criteria for the test. For regression testing, this can be a reference log file from a previous run; the test passes if the new behavior (e.g., trajectory) is sufficiently similar to the reference.
- **Execution Agent (Optional):** Defines how and where the test is run, specifying the execution engine (e.g., local, Docker, or Kubernetes) and the number of (parallel) repetitions to account for non-determinism.

A sample test description for a PX4 drone is shown in Listing 3.1. This single, self-contained file provides all the information Aerialist needs to autonomously execute, analyze, and report on the test.

```yaml
# 1. Robot Configuration
robot:
  type: px4-sitl
  params_file: path/to/navigation_params.csv

# 2. Environment Configuration
simulation:
  simulator: gazebo
  headless: true
  obstacles:
    - shape: box
      size: {l:10, w:2, h:8}
      position: {x:20, y:15, z:0, r:0}
    - shape: cylinder
      size: {r:1, h:10}
      position: {x:25, y:-10, z:0, r:0}

# 3. Mission Configuration
mission:
  commands_file: path/to/mission_commands.csv
  mission_file: path/to/survey_mission.plan

# 4. Expectations
assertion:
  log_file: path/to/golden_run.ulg # Optional: for
   regression testing
  variable: trajectory

# 5. Execution Agent
agent:
  engine: kubernetes # {local, docker, kubernetes}
  count: 10 # Number of parallel runs to check for non-
   determinism
```

Listing 3.1. A sample Aerialist test description in YAML format.

3.3.2 Architecture and Capabilities

The Aerialist framework is architected around a central **Test Runner** subsystem, which ingests the YAML test description and orchestrates the entire testing process using a set of specialized, modular components (Figure 3.3).

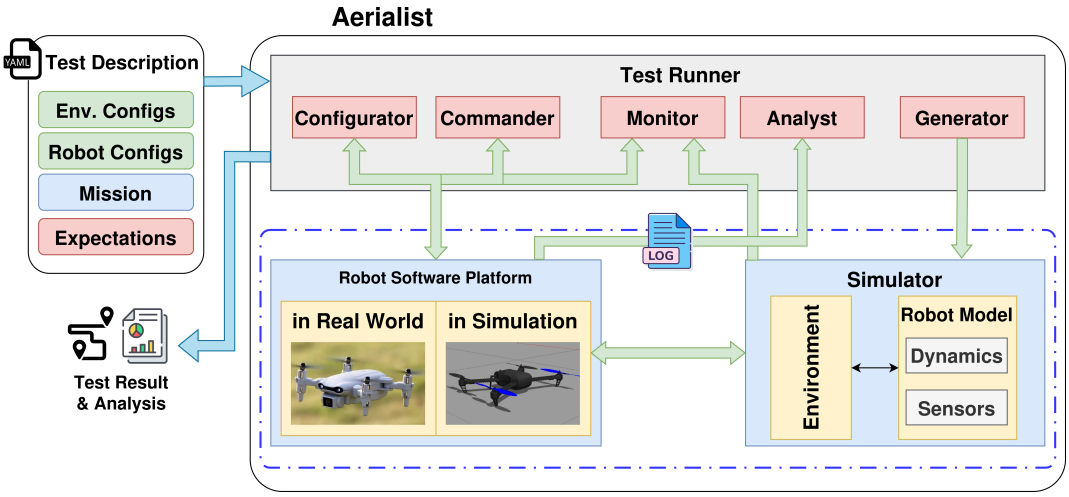

Figure 3.3. The high-level architecture of the Aerialist framework, showcasing its modular design and interaction with the robot's software stack and the simulator [59].

The **Generator** module first parses the 'simulation' block of the test description to set up the specified virtual world, placing the robot and any obstacles. The **Configurator** then uses the 'robot' configuration to initialize the robot under test, setting its parameters and establishing communication. During the test run, the **Commander** triggers the autonomous missions and sends any scheduled inputs defined in the 'mission' block, while the **Monitor** observes the robot's and simulators' states, checking for seamless executions of the tests and handling any runtime issues.

A key capability of the framework is its ability to manage large-scale experiments. By interpreting the 'agent' block in the test description, the Test Runner can distribute multiple simulation runs as isolated jobs across a **Kubernetes** cluster. This **scalable and parallel execution** is essential for modern search-based testing techniques that would otherwise be prohibitively time-consuming.

After execution, the **Analyst** module handles all post-mortem processing. It automatically parses the detailed logs (e.g., PX4 'ulog' files or ROS 'bag' files) and classifies the final outcome into predefined categories such as *Success*, *Timeout*, or *Failure*. This automated classification is essential for efficiently calculating high-level metrics like mission success rate across a large test suite. To address the challenge of non-determinism in robotic systems, the Analyst module can aggregate results from multiple parallel runs, providing a statistically robust assessment of system behavior by reporting the distribution of outcomes or computing an average trajectory.

Perhaps the most critical feature of the Analyst module is its ability to automatically generate insightful visualizations from the raw log data. As shown in Figure 3.4, these

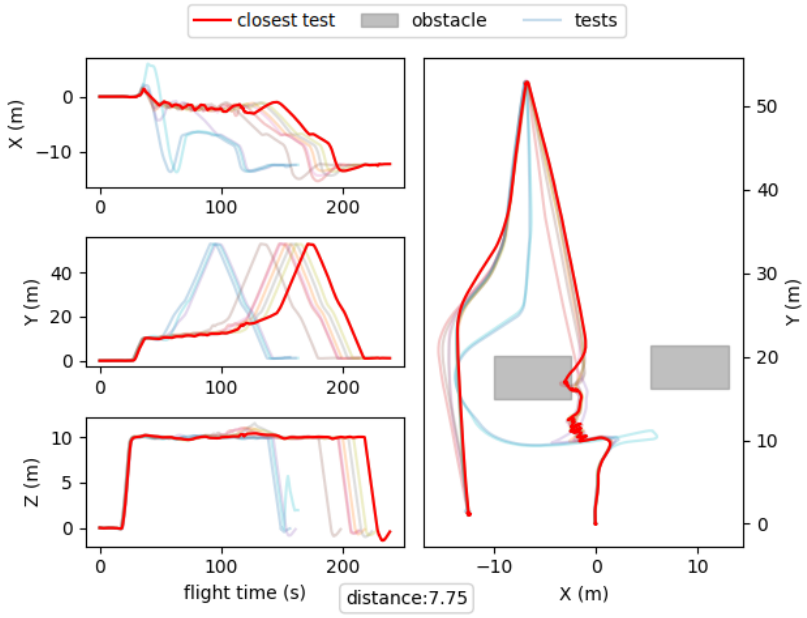

Figure 3.4. An example of an automated visualization generated by Aerialist's Analyst module. The plot shows 10 repeated runs of a single, "flaky" test case, immediately revealing the non-deterministic behavior of the UAV's navigation algorithm under challenging conditions [59].

plots provide an immediate and intuitive understanding of the robot's behavior during a test. They typically include a 2D top-down view of the robot's trajectory overlaid on the environment's obstacle layout, as well as time-series graphs of key state variables like position, orientation, and velocity. For the tests executed multiple times in parallel, they also include an average trajectory to help identify the "common" behavior. For developers, these visualizations are an invaluable debugging tool. They make it possible to quickly diagnose the root cause of a failure—for instance, by observing exactly where a robot deviated from its intended path or exhibited erratic behavior—without needing to manually parse complex, low-level log files. This capability significantly accelerates the debugging and development cycle and was highlighted as a significant improvement over existing tools during the industrial evaluation at ANYbotics (detailed in Chapter 5).

3.3.3 Chapter Summary

This chapter established the significant engineering challenges inherent in the verification of autonomous robotic systems, rooted in the complexity of robotic mobility. We introduced two distinct and demanding use cases—an autonomous UAV and an indus-

trial quadruped—that exemplify these challenges and serve as the validation platforms for this dissertation.

As a foundational contribution, we presented **Aerialist**, a general-purpose, extensible framework for automating simulation-based testing of complex robotic systems. By leveraging a standardized test description model, a modular architecture, and advanced capabilities for scalable execution and automated analysis, Aerialist provides a powerful open-source[3] platform that overcomes many of the primary obstacles to rigorous robotics testing. The practical value of these capabilities, particularly its usability and powerful visualizations, was strongly validated during its industrial adoption at ANYbotics discussed in detail in Chapter 5. This framework provides the essential tooling that underpins the advanced research on test generation and runtime monitoring presented in the chapters that follow.

[3] https://github.com/skhatiri/Aerialist

Chapter 4

Realistic Simulation-based Test Generation for Robotic Navigation Systems

The previous chapter introduced *Aerialist*, a foundational framework for automating the execution and analysis of tests for complex robotic systems. While Aerialist provides the essential "how"—the infrastructure to run tests in a scalable and repeatable manner—it does not address the equally critical question of "what" to test. The effectiveness of any testing effort is ultimately determined by the quality of the test cases themselves. For autonomous robotic systems, a high-quality test case is one that is both representative of real-world conditions and challenging enough to uncover safety-critical flaws in the robot's navigation logic.

This chapter presents *Surrealist*, a novel, search-based methodology designed to automatically generate such high-quality test cases [58]. We begin by framing the central problem: the "reality gap" that limits the effectiveness of conventional simulation-based testing. We then introduce the generic, two-phase approach of Surrealist, which addresses this gap by first replicating real-world behaviors in simulation and then generating challenging variants in the neighborhood of this replicated reality. The chapter concludes with a detailed case study demonstrating the successful application of this methodology to the domain of Unmanned Aerial Vehicles (UAVs), validating its ability to produce realistic and failure-inducing test scenarios.

4.1 The Simulation-to-Reality Gap in Robotic Testing

As established in the State of the Art, simulation-based testing represents a fundamental practice for verifying the safety and reliability of autonomous robotic systems [3,

108]. It offers a cheaper, safer, and more scalable alternative to physical field testing. However, its practical utility is often limited because the test scenarios considered in a simulator may not be representative of the actual operational scenarios the robot will experience in the field. This discrepancy makes it challenging to capture the same bugs in simulation that might occur in the physical world [4, 109] and to generate simulated test cases that can reliably expose realistic failures [3].

To illustrate this problem, consider a common scenario in robotics development. A robotics engineer is tasked with verifying the proper functionality and safety of a drone's obstacle avoidance system. The drone has been successfully tested in a real-world environment containing a few obstacles, and the operational data, or *flight logs*, from this test are available. The engineer knows that the system worked in that specific configuration, but has no guarantee that it will behave safely under slightly different conditions, such as different obstacle placements, sizes, or orientations. Manually testing all such variations in the field is infeasible due to time and budget constraints.

The logical next step is to use a simulator to explore these variations. However, this immediately presents two significant challenges. First, how can the engineer faithfully replicate the original real-world test in the simulation? The physical attributes of the simulated and real drone are never identical, and unknown environmental factors can influence behavior. Simply replaying the same commands will not result in the same trajectory. Second, once a replication is achieved, how can the engineer efficiently explore the vast space of possible variations to find those few critical scenarios that might trigger an unsafe behavior? Manually designing these "corner cases" is difficult and often relies on intuition rather than systematic exploration.

This highlights the need for a new approach to test generation—one that can automatically bridge the reality gap by grounding simulated tests in real-world data and then systematically searching for challenging scenarios. While some automated solutions exist for reproducing software crashes [11, 84, 99], they do not address the complexity of robotic systems, where the state to be reproduced involves not just the program's internal state, but also the dynamic state of the physical world.

4.2 Surrealist: A Two-Phase Approach for Test Generation

To address the challenges of realism and test effectiveness, this thesis introduces **Surrealist**[1], a novel, search-based methodology for automatically generating high-quality test cases for robotic systems [58]. The core idea is to bridge the reality gap by ground-

[1] https://github.com/skhatiri/Surrealist

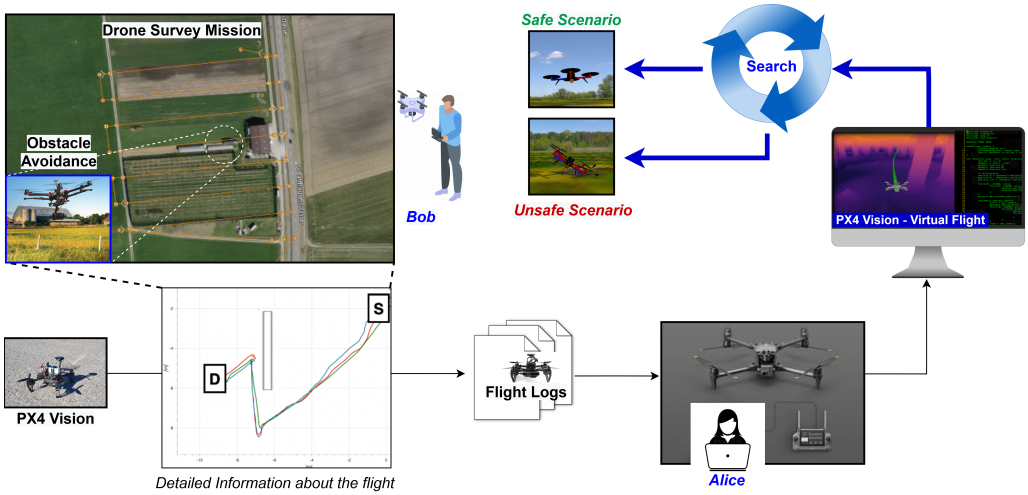

Figure 4.1. Example Surrealist workflow: starting from real-world flight data, the framework reconstructs a faithful simulated counterpart and systematically explores hundreds of realistic variations to uncover potential safety issues.

ing the test generation process in data from real-world operations. Surrealist accomplishes this through a unique, two-phase approach that explicitly separates the goal of achieving realism from the goal of finding failures.

4.2.1 Phase 1: Realistic Test Replication from Field Data

The first phase, **Realistic Test Replication**, aims to faithfully reproduce a robot's observed behavior from a physical field test within the simulation environment. Given an operational log file from a real-world run, Surrealist treats the robot's recorded trajectory and sensor data as the ground truth. It then formulates an optimization problem: to find the optimal configuration of the simulated environment (e.g.,, the unknown positions and sizes of obstacles) that minimizes a distance measure between the simulated behavior and the real-world behavior. The output of this phase is a high-fidelity simulated test case that mirrors a known, real-world scenario, effectively bridging the reality gap for a specific operational context.

4.2.2 Phase 2: Generating Challenging Scenarios

The second phase, **Challenging Scenario Generation**, leverages the realistic test case from the first phase as a validated starting point. Instead of seeking similarity to reality, the goal now shifts to discovering safety-critical edge cases. Starting from the replicated scenario, Surrealist smoothly manipulates the environment and mission properties to

generate variants in the “neighborhood” of the original test (e.g., by moving, resizing, or rotating an obstacle). The objective is to find test cases that are more challenging for the robot’s navigation system, pushing it to its operational limits and potentially triggering unsafe behaviors or latent bugs that would be difficult to discover through manual design or purely random testing.

4.2.3 The Generic Search-Based Algorithm

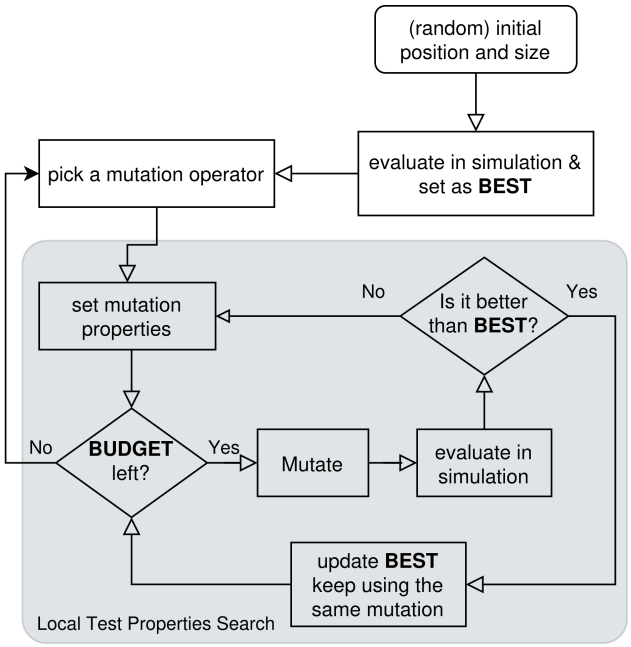

Figure 4.2. High-level flowchart of the generic search-based optimization algorithm.

Both phases of the Surrealist methodology are powered by the generic, search-based optimization algorithm depicted in Figure 4.2. The problem is formulated as finding an optimal set of test properties that maximizes a given *fitness function*, which is tailored to the specific goal of each phase (i.e., realism or challenge).

As shown in the flowchart, the search process is an iterative, **adaptive greedy algorithm**. It begins by evaluating an initial *seed solution* (e.g., a random or manually defined obstacle configuration) and setting it as the current ‘BEST‘ solution. The algorithm then enters its main loop, which systematically explores the solution space by applying parameterized operators called *mutators* to individual test properties (e.g., moving an obstacle or a waypoint).

The core of the process is a local search loop. The algorithm first picks a mutation operator and then repeatedly applies it to the ‘BEST‘ solution to generate a new

candidate. This new candidate is evaluated in simulation. If its fitness is better than the current 'BEST', the solution is updated, and the algorithm continues to explore in that promising direction. If the new candidate is not better, the local search adapts by adjusting its mutation properties (e.g., reducing the step size) to explore the area more finely. This local search for a given mutator continues until its allocated simulation 'BUDGET' is exhausted or no further improvements are found. The algorithm then picks the next mutation operator and repeats the process, terminating when the global budget is spent.

Each candidate solution (a set of test properties) is evaluated by automatically executing the corresponding test case using the Aerialist framework. To account for the non-deterministic nature of robotic systems, each test case can be executed multiple times in parallel, with the results being aggregated to produce a reliable fitness score. The *fitness function* is the core component that guides the search. For the replication phase, the fitness is based on a distance metric that quantifies the dissimilarity between the simulated and real-world trajectories. For the challenge generation phase, the fitness function measures the risk or difficulty of the scenario, for instance, by calculating the minimum distance between the robot and any obstacles during the mission.

4.3 Instantiating the Framework for Obstacle Avoidance

The generic, two-phase methodology of Surrealist can be instantiated for a wide range of robotics testing problems. This section details its specific application to the critical and ubiquitous challenge of autonomous navigation and obstacle avoidance. To apply the framework, one must define the concrete test properties to be optimized, the mutation operators that manipulate them, and the specific fitness functions that guide the search in each phase.

4.3.1 Test Properties and Mutation Operators

For a typical navigation task, the search space is defined by two key categories of tunable parameters: the configuration of the environment and the specification of the robot's mission. In this instantiation, Surrealist considers the properties of the **obstacles** in the world and the **waypoints** that define the robot's path.

The environmental parameters are defined by the physical properties of each of the N obstacles, including its **size** (e.g., length, width, and height for a box; or radius and height for a cylinder), its **position** as (x, y, z) coordinates, and its **orientation** in the vertical axis (yaw). Complementing this, the mission is defined by a sequence of waypoints, each with a target **position** (x, y, z) and desired **orientation**.

To explore this combined search space, Surrealist employs a set of simple yet effective **mutation operators** that can be applied to individual obstacles or waypoints:

- **Move Obstacle:** Translates an obstacle from its current location by Δ_x and Δ_y.
- **Resize Obstacle:** Modifies an obstacle's dimensions (e.g., its length, width, or radius) while keeping its center position fixed.
- **Rotate Obstacle:** Rotates an obstacle around its geometric center by an angle of Δ_r.
- **Move Waypoint:** Translates a specific waypoint in the mission plan by Δ_{wp-x}, Δ_{wp-y}, or Δ_{wp-z} from the current coordinates.
- **Rotate Waypoint:** Rotates the desired orientation of a specific waypoint by an angle of Δ_{wp-r}.

The search algorithm, detailed below, systematically applies these mutators to find the optimal obstacle configurations according to the fitness function of the current phase.

4.3.2 Fitness Functions

The direction of the search is guided entirely by the fitness function, which must be carefully defined to match the goal of each phase.

Fitness for Realistic Test Replication

In the first phase, the goal is to minimize the difference between the robot's trajectory in simulation and its trajectory from a real-world log file. As these trajectories are multi-dimensional time series of potentially different lengths, a specialized distance metric is required. For this purpose, we use **Dynamic Time Warping (DTW)** [14], a well-established algorithm for measuring similarity between two temporal sequences. DTW finds the optimal alignment between two trajectories by minimizing the Euclidean distance between their corresponding points. The recursive formula for DTW is:

$$DTW[i,j] = d(s_i, t_j) + \min\{DTW[i-1,j], DTW[i,j-1], DTW[i-1,j-1]\} \tag{4.1}$$

where $d(s_i, t_j)$ is the Euclidean distance between point i of the simulated trajectory s and point j of the real-world trajectory t. To handle the non-determinism of multiple simulation runs, we first compute an average trajectory from all runs using *DTW Barycenter Averaging* [89] before comparing it to the real-world log. Since the search

algorithm is designed to maximize fitness, the final fitness value is the negation of the DTW distance ($-DTW$).

Fitness for Challenging Scenario Generation

In the second phase, the objective shifts from realism to finding safety-critical scenarios. The goal is to generate test cases that force the robot to navigate in close proximity to obstacles. The fitness function is therefore designed to reward configurations that result in low robot-to-obstacle distances. It consists of two components:

$$\begin{aligned} sum\_dist &= \min_{p \in \text{trj.points}} \sum_{o \in \text{obs}} d(p, o) \\ min\_dist &= \min_{o \in \text{obs}, p \in \text{trj.points}} d(p, o) \\ fitness &= -(sum\_dist + 2 \times min\_dist) \end{aligned} \tag{4.2}$$

Here, *sum_dist* encourages configurations where obstacles are generally close to the robot's path, while *min_dist* specifically rewards scenarios where the robot gets very close to a single obstacle. The *min_dist* component is given a higher weight as it is a stronger indicator of a potentially risky scenario. The entire expression is negated to be maximized by the search algorithm.

4.3.3 The Search Algorithm in Detail

The adaptive greedy search algorithm that powers both phases is presented in Algorithm 1 and Algorithm 2. Algorithm 1 orchestrates the high-level search, iterating through all available mutators and invoking a local search procedure for each one. It manages the overall simulation budget and terminates when no further improvements can be found.

Algorithm 2 implements the core optimization logic. It performs a greedy search on the parameter space of a single mutator (e.g.,, the distance to move an obstacle). At each step, it evaluates two new solutions by moving in positive and negative directions from the current best parameter value. If an improvement is found, it accepts the new solution and continues in that direction. To accelerate convergence, the step size is doubled if the search consistently moves in the same direction. If no improvement is found, the step size is halved, allowing for a finer-grained search around the local optimum. This adaptive process allows the algorithm to quickly make large improvements when far from an optimum and to perform a precise search when close to one, making efficient use of the simulation budget.

Algorithm 1: GENERIC-TEST-PROPERTIES-SEARCH

```
Input : seed: original test properties
        fitness_func: fitness function to maximize
        budget: global search budget (max num of simulations)
        min_rounds: minimum ensured round of global search
Result: best: test properties that optimize the fitness
1  begin
2      best = seed
3      evaluation_hash = {}
4      EVALUATE(fitness_func, best, evaluation_hash, budget, NULL)
5      improved = true
6      while improved do
7          improved = false
8          local_budget = budget/(|seed.mutators|×min_rounds)
9          for mutator in seed.mutators do
10             improved = improved ∨ LOCAL-TEST-PROPERTIES-SEARCH(best,
                mutator, evaluation_hash, local_budget, budget)
11         min_rounds = min_rounds - 1
12     return best
```

4.4 Empirical Validation for Autonomous UAVs

To empirically validate the generic, two-phase methodology of Surrealist, the framework was first implemented and rigorously evaluated in the complex and safety-critical domain of autonomous UAVs [58]. This case study, leveraging the Aerialist framework for all test automation and execution, was designed to provide strong evidence for the approach's effectiveness by answering two primary research questions, each corresponding to one of the core phases of Surrealist.

- **RQ_1 (Flight Replication):** *Can Surrealist generate simulated test cases that faithfully replicate autonomous flight trajectories from real-world flight logs?* The goal was to replicate the test environment such that the simulated UAV's trajectory was as similar as possible to the original one, using only the logged data as input.

- **RQ_2 (Test Generation):** *Can Surrealist modify a replicated, simulation-based test case to make it more challenging for the UAV's autonomous controller?* The goal was to investigate whether the framework could, starting from a realistic baseline, discover new environmental configurations that result in unsafe or faulty behavior.

Algorithm 2: LOCAL-TEST-PROPERTIES-SEARCH

InOut : best: best solution found so far
Input : mutator: test property mutator
InOut : evaluation_hash: memory of past evaluations
Input : local_budget: max simulations for current mutator
InOut : budget: overall max simulations allowed
Result: improved: previous best solution was improved

```
1  begin
2      step = mutator.default_step
3      param = mutator.default_value
4      positive_moves = negative_moves = 0
5      iter_with_no_improvements = 0
6      new_best = best
7      while local_budget > 0 ∧ iter_with_no_improvements < MAX_IT do
8          mu1 = MUTATE(best, mutator, param + step)
9          mu2 = MUTATE(best, mutator, param - step)
10         EVALUATE(mu1, evaluation_hash, budget, local_budget)
11         EVALUATE(mu2, evaluation_hash, budget, local_budget)
12         if mu1.fitness > new_best.fitness + ϵ ∧ mu1.fitness > mu2.fitneess then
13             new_best = mu1
14             param = param + step
15             positive_moves += 1
16             negative_moves = 0
17             iter_with_no_improvements = 0
18             if positive_moves > MAX_SEQ_IT then
19                 step = step · 2
20         else if mu2.fitness > new_best.fitness + ϵ then
21             new_best = mu2
22             param = param - step
23             negative_moves += 1
24             positive_moves = 0
25             iter_with_no_improvements = 0
26             if negative_moves > MAX_SEQ_IT then
27                 step = step · 2
28         else
29             if |new_best.fitness - mu1.fitness| < ϵ ∧ |new_best.fitness - mu2.fitness| < ϵ
                 then
30                 break
31             step = step / 2
32             positive_moves = negative_moves = 0
33             iter_with_no_improvements += 1
34     improved = false
35     if new_best ≠ best then
36         best = new_best
37         improved = true
38     return improved
```

4.4.1 Experimental Design

The validation was structured as a controlled experiment aimed at assessing Surrealist's ability to both replicate real-world flights and generate challenging new scenarios that expose system weaknesses.

System Under Test and Scenarios

The subject of the experiments was a simulated quadcopter controlled by the **PX4 Autopilot** software stack [80], with the **PX4-Avoidance** module [91] enabled for autonomous navigation. All experiments were conducted in the **Gazebo** simulator [64].

For $\mathbf{RQ}_1$, the ground truth was an original flight conducted in a real-world environment. A UAV was flown on an autonomous mission in an empty outdoor parking area containing a large cargo container (sized approximately $2.5m \times 12m \times 2.5m$). The trajectory from this physical test, captured in a flight log [57], served as the target for replication.

For $\mathbf{RQ}_2$, a baseline simulated test case was created. In this scenario, a UAV performed a mission that required it to safely navigate around a single, large, building-like obstacle ($8m \times 5m \times 20m$). The objective for Surrealist was to start from this safe scenario and generate a more challenging version by introducing and optimizing the position of a second obstacle.

Experimental Procedure and Metrics

To run our experiments, we set the hyper-parameters of Algorithms 1,2 (see Section 4.3.3) to the values reported in Table 4.1. To evaluate our approach, we run SURREALIST to replicate the flight trajectory (RQ_1) or generate test cases (RQ_2), applying 10 repetitions with the same configurations, to gain statistical validity of our results. To manage the non-determinism of the simulator, each individual test case evaluation within the search process involved multiple parallel simulations (5 for RQ_1, 10 for RQ_2). The entire experimental campaign was executed on a Kubernetes cluster, orchestrated by the Aerialist framework.

The effectiveness of the approach was measured using several metrics reported in Table 4.2. For $\mathbf{RQ}_1$, the primary metrics were the reduction in **Dynamic Time Warping (DTW)** distance and **Fréchet distance** [28] between the simulated and real-world trajectories, comparing the initial seed environment to the final, optimized environment. For $\mathbf{RQ}_2$, the metrics included the reduction in the **minimum distance** between the UAV and any obstacle, as well as the rate of **crashes** and **unsafe behaviors** (defined as flying closer than 1.5m to an obstacle) in the final generated test case. The statistical

Table 4.1. Experiment Hyper-parameters

RQ	Parameter	Value
$RQ_{1,2}$	repetition	10
$RQ_{1,2}$	MAX_SEQ.	5
$RQ_{1,2}$	MAX_IT	5
$RQ_{1,2}$	default_step	4m (MOVE, RESIZE), 30° (ROTATE)
$RQ_{1,2}$	default_value	0m (MOVE, RESIZE), 0° (ROTATE)
$RQ_{1,2}$	ϵ	0
RQ_1	budget	200 (seed 1), 100 (seed 2)
RQ_1	min_rounds	4 (seed 1), 2 (seed 2)
RQ_1	sim. runs	5
RQ_2	budget	50
RQ_2	min_rounds	2
RQ_2	sim. runs	10

Table 4.2. Experiment Evaluation Metrics

Metric	Definition
DTW Reduction (RQ_1)	$= 1 - \frac{DTW(best,org)}{DTW(seed,org)}(\%)$
Fréchet Reduction (RQ_1)	$= 1 - \frac{Frechet(best,org)}{Frechet(seed,org)}(\%)$
Min_dist Red. (RQ_2)	$= 1 - \frac{Min\_dist(best)}{Min\_dist(seed)}(\%)$
Crash & Unsafe Rate (RQ_2)	= % of crash & unsafe in the simulations of best
P-value	comparing seed and best fitness distributions
Effect Size	comparing seed and best fitness distributions
Needed Budget	= # of evaluations to reach best
Eval. Time	= average time for each solution evaluation

significance of the results was verified using a one-way ANOVA test, and the magnitude of the improvement was quantified using the Cohen's d effect size statistic.

4.4.2 Results: Faithful Replication of a Real-World Flight (RQ_1)

The first phase of the evaluation confirmed that Surrealist can successfully bridge the reality gap by faithfully replicating a real-world flight in simulation. The search started from two different seed configurations as plotted in Figure 4.3, each with a small, generic obstacle placed on opposite sides of the flight path to ensure the search was

not dependent on the starting point.

As shown in Figure 4.4, Surrealist was consistently able to find an optimal size, position, and rotation for the simulated obstacle that caused the UAV's trajectory to closely match the one from the real-world log. Although the final obstacle properties found in each of the 10 repetitions were not identical, the resulting flight behaviors were remarkably consistent and realistic. For example, in one representative run, the algorithm discovered that moving the obstacle 2 meters upwards, rotating it 30 degrees, and increasing its height by 2.8 meters led to a near-perfect replication.

The quantitative results, summarized in Table 4.3, underscore the significance of this achievement. For the first seed, Surrealist reduced the average DTW distance by **83.1%** (from 383.5 to 63.15) and the average Fréchet distance by **83.4%** (from 6.87m to 1.14m). For the second seed, the reductions were 48.8% and 51.6% respectively. The improvements were statistically significant with a very large effect size.

Notably, the final DTW values achieved by the search (63.15 and 59.1) were very close to the inherent non-determinism of the simulator itself, which was estimated to have a baseline DTW of 58.5 between identical runs. This indicates that the replication was about as accurate as theoretically possible, demonstrating that the information available in operational logs is sufficient for Surrealist to find test properties that faithfully replicate autonomous flight trajectories in simulation.

Table 4.3. RQ_1 Evaluation Metrics: A comparison of trajectory similarity before (Seed) and after (Best) optimization by Surrealist, averaged over 10 repetitions.

Metric	**Seed 1**	**Seed 2**
Seed DTW	383.5	116.1
Best DTW	63.15	59.1
DTW Reduction (%)	**83.1%**	**48.8%**
Seed Fréchet (m)	6.87	2.25
Best Fréchet (m)	1.14	1.05
Fréchet Reduction (%)	**83.4%**	**51.6%**
P-value	1.9e-12	1.2e-12
Effect Size (Cohen's d)	7.9 (Large)	8.13 (Large)
Required Simulation Budget	74.8	65
Average Simulation Time	152.9 (s)	158.9 (s)

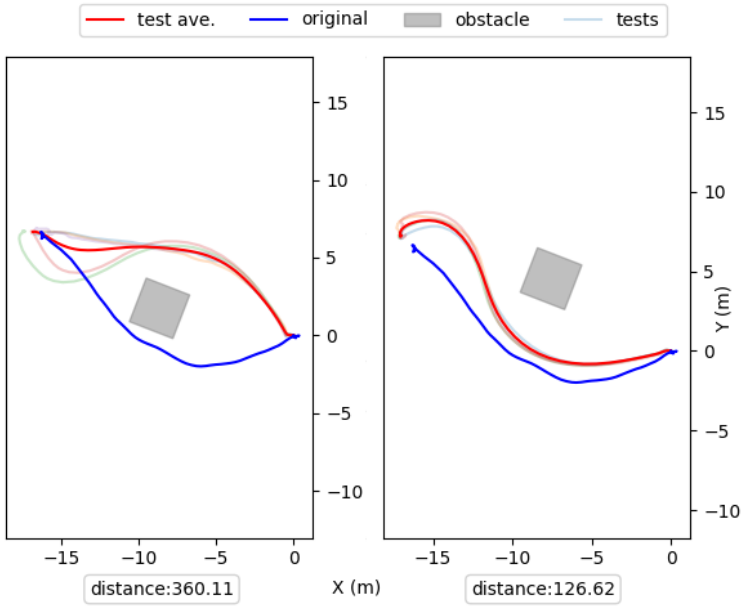

Figure 4.3. RQ_1 Seeds: Two different obstacle placements used as the seed scenario for the test generation process and the corresponding flight trajectories (red) [58].

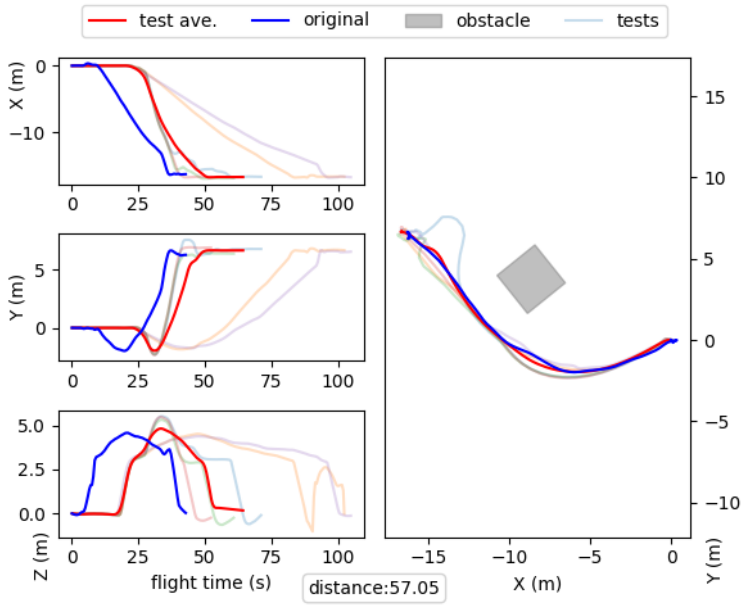

Figure 4.4. RQ_1 Results: A comparison of the real-world flight trajectory (blue) and the faithfully replicated trajectory from the best-found simulation (red). Surrealist automatically discovered the optimal obstacle configuration to achieve this high-fidelity replication [58]. All the test generation steps and corresponding plots can be found in the replication package [56].

Finding 1 (Flight Replication): The Surrealist framework successfully bridged the reality gap by automatically discovering environmental configurations that faithfully replicated a real-world UAV flight in simulation. The search-based approach significantly reduced trajectory dissimilarity (up to 83.4% reduction in Fréchet distance), achieving a level of fidelity close to the inherent non-determinism of the simulator itself.

4.4.3 Results: Generation of Unsafe and Crashing Scenarios (RQ_2)

The second phase of the evaluation demonstrated that, starting from a realistic and safe baseline, Surrealist can effectively generate novel test cases that expose critical safety flaws. The search began with a safe scenario where the UAV reliably navigated around a single large obstacle. By introducing a second obstacle as illustrated in Figure 4.5 and using the risk-based fitness function, Surrealist discovered multiple challenging environmental configurations that consistently triggered unsafe behavior.

The final generated test case, shown in Figure 4.6, revealed a latent bug in the UAV's navigation logic. In this scenario, the UAV becomes "confused" and hesitates, leading to highly non-deterministic and dangerous trajectories. Across 10 repeated runs of this single, automatically generated test case, the UAV **crashed** into an obstacle in **25%** of the runs and violated the 1.5m safety margin in **84%** of the runs.

Table 4.4. RQ_2 Evaluation Metrics: A comparison of the safety risk before (Seed) and after (Best) optimization by Surrealist.

Metric	**Average Value**
Seed Min Distance (m)	3.36
Best Min Distance (m)	0
Min Distance Reduction (%)	**100%**
Crash Rate (%)	25%
Unsafe Rate (%)	84%
P-value	6.7e-12
Effect Size (Cohen's d)	119.9 (Large)
Required Simulation Budget	36.8
Average Simulation Time	388.1 (s)

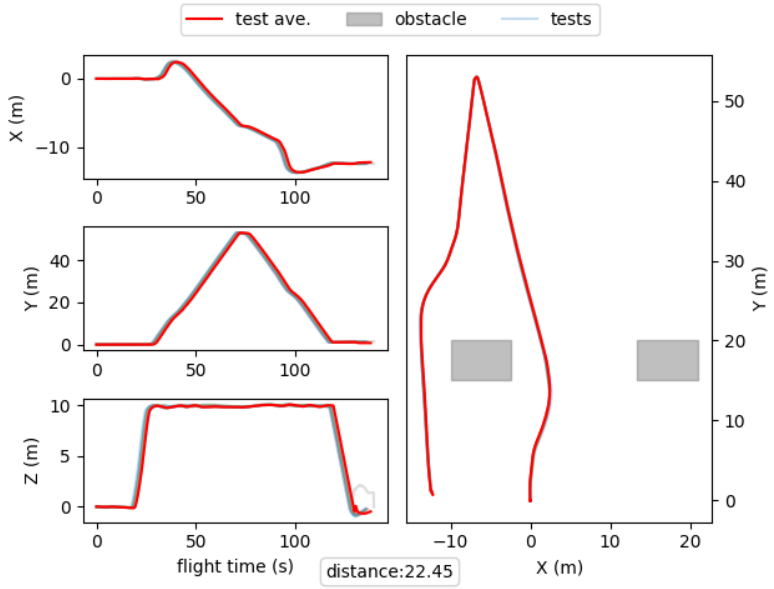

Figure 4.5. RQ_2 Seed: The additional obstacle (right) with the same size as the original one is placed far enough not to influence the flight trajectory [58].

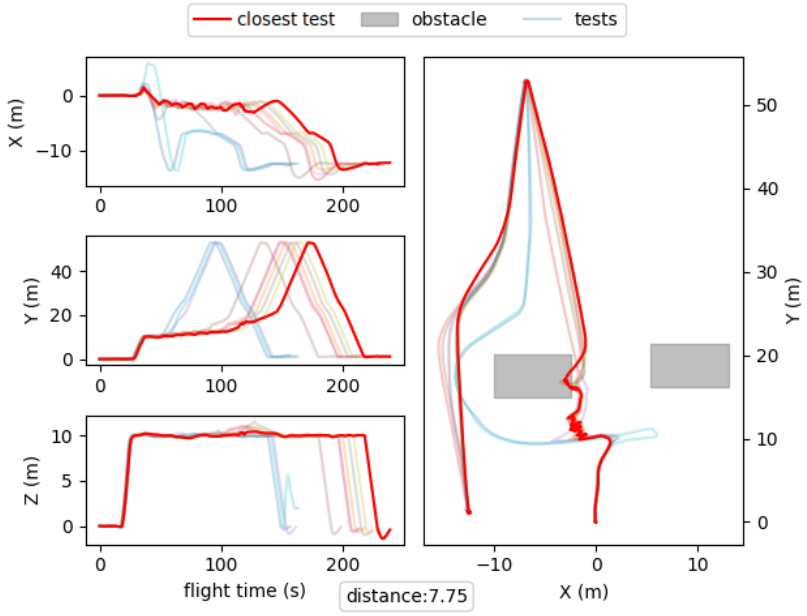

Figure 4.6. RQ_2 Results: A challenging and flaky test case automatically generated by Surrealist. The 10 different colored lines represent 10 runs of this single scenario, revealing highly non-deterministic behavior and multiple crashes [58]. All the test generation steps and corresponding plots can be found in the replication package [56].

The quantitative results, shown in Table 4.4, confirm the effectiveness of the search. The algorithm successfully reduced the minimum distance to the obstacle by **100%** (from a safe 3.36m in the seed scenario to 0m in the final test). This result was also statistically significant with an extremely large effect size.

This finding demonstrates that by systematically modifying a simulation-based test case, Surrealist can generate challenging scenarios that expose the UAV to unsafe behaviors or even crashes, successfully identifying safety-critical corner cases that would be difficult to find through manual test design.

Finding 2 (Test Generation): Starting from a safe, replicated scenario, Surrealist effectively generated challenging new test cases that exposed a latent bug in the UAV's navigation logic. The automatically generated scenario consistently induced unsafe and non-deterministic behaviors, resulting in a 25% crash rate and an 84% safety violation rate across repeated runs.

4.4.4 Case Study Conclusion and Limitations

This case study provides strong empirical evidence for the effectiveness of the Surrealist methodology in the demanding domain of aerial robotics. The experiments successfully demonstrax'ted that the two-phase approach can both bridge the reality gap by replicating real-world behaviors and effectively generate challenging test scenarios that uncover safety-critical failures. While these results are promising, it is important to consider the threats to validity inherent in such an empirical study.

The conclusions regarding the search algorithm's effectiveness are supported by appropriate statistical tests, confirming that the observed improvements are significant with a large effect size (**conclusion validity**). However, the choice of metrics is crucial for evaluating the results. The study relied on Dynamic Time Warping (DTW) and Fréchet distance for measuring trajectory similarity—well-established metrics, but not the only possible choices (**construct validity**). Furthermore, the inherent non-determinism of complex simulations was a key challenge, which was mitigated by leveraging the Aerialist framework to execute each test case multiple times and aggregate the results.

The experimental results are also specific to the technologies used, namely the Gazebo simulator and the PX4 software stack. While these are widely adopted, further studies would be needed to confirm the findings on other platforms (**internal validity**). Similarly, while multiple seed scenarios were used, the vastness of the search space means that different starting points could lead to different outcomes.

Finally, and most importantly, the primary validation in this chapter was conducted on a single, albeit complex, robotic domain: UAVs. While the Surrealist methodology is

designed to be generic, these results alone are not sufficient to claim its universal applicability (**external validity**). The question of whether this approach can be successfully generalized to other platforms with different locomotion and navigation challenges remains open. Addressing this crucial question of industrial adoption and cross-domain validation is the primary focus of the next chapter.

4.5 Benchmarking Test Generation Techniques: The UAV Testing Competition

To further validate the research presented in this dissertation and to inspire broader engagement from the software testing community, the frameworks developed as part of this work—Aerialist and Surrealist—were used to establish the first international **UAV Testing Competition**[2] [60, 61, 62]. The competition provides a standardized platform for researchers to develop and benchmark their own test generation tools for a complex, vision-based autonomous flight system. The core objective for participants is to generate diverse and effective test suites that uncover critical vulnerabilities in the PX4 obstacle avoidance system by manipulating the size and placement of obstacles in the simulation environment.

The competition is built directly upon the contributions of this thesis. The **Aerialist** framework serves as the official benchmarking platform, providing all participants with a robust and easy-to-use infrastructure for defining and executing their tests. The original **Surrealist** tool, with its adaptive greedy search, was established as the state-of-the-art baseline against which all new tool submissions are evaluated.

4.5.1 Competition Format and Evolution

Over its first two editions in 2024 and 2025, held at the International Workshop on Search-Based and Fuzz Testing (SBFT@ICSE) and the International Conference on Software Testing, Verification and Validation (ICST), the competition has attracted significant interest, with a total of 11 unique and novel tool submissions. The rules and evaluation criteria have evolved to reflect a growing maturity and a deeper understanding of the challenges in UAV testing.

In the inaugural **2024 edition**, participants were tasked with generating test suites by adding up to four box-shaped obstacles to a set of predefined mission scenarios [60]. The evaluation was based on a single, comprehensive scoring formula that rewarded the discovery of severe failures (i.e., flights with a low minimum distance to an ob-

[2] https://github.com/skhatiri/UAV-Testing-Competition/

stacle), while also considering test case complexity and diversity. A test's score was calculated based on the minimum distance observed across 5 repeated simulations:

$$point(sim) = \begin{cases} 5, & \text{if } min\_dist(sim) < 0.25m \\ 2, & \text{if } 0.25m \leq min\_dist(sim) < 1m \\ 1, & \text{if } 1m \leq min\_dist(sim) < 1.5m \\ 0, & \text{if } min\_dist(sim) \geq 1.5m \end{cases} \tag{4.3}$$

For the **2025 edition**, the rules were refined to increase the challenge and improve the evaluation process. The maximum number of obstacles was reduced to three, and a test suite size limit of 20 was introduced, rewarding tools that could effectively prioritize their most critical test cases. Most importantly, the evaluation was split into two distinct metrics: a **Failure Score** to measure effectiveness at finding bugs, and a **Diversity Score** to measure the geometric dissimilarity of the generated environments [61, 62]. The initiative is ongoing, with the **2026 edition** planned to introduce even more complex challenges, such as environmental factors like wind, continuing to push the boundaries of automated testing for autonomous systems.

4.5.2 A Showcase of Advanced Testing Techniques

The competition has served as a platform for a wide range of advanced test generation techniques, highlighting the innovation in the field. The tools submitted in the first two editions include:

- **Evolutionary and Search-Based Algorithms:** Several tools employed advanced search metaheuristics, including a (1+1) Evolutionary Algorithm (**Evolv-1** [66]), multi-objective search (**OptObstacles** [51]), Illumination search (**DeepHyperion-UAV** [121]), and a cost-aware, mutation-based algorithm (**CAMBA** [69]).

- **Tree-Search Algorithms:** Two tools, **TUMB** [104] and its successor **PALM** [105], utilized Monte Carlo Tree Search (MCTS) to explore the vast search space of obstacle placements.

- **Surrogate Model-Based Approaches:** To reduce the high cost of simulation, some tools employed surrogate models. These include using an RRT* path planning algorithm as a fast surrogate fitness function (**AmbieGen** [45]), and using predictions from a lower-fidelity simulator to guide test selection in the full-fidelity simulation (**Pseudo-Random** [96]).

- **Machine Learning and Learning-Based Approaches:** Other participants leveraged machine learning, including generative adversarial networks (**WOGAN-UAV** [110]), Q-learning with an Upper Confidence Bound strategy (**TGen-UQ** [50]), and even Large Language Models (**TAIiST** [120]).

The results across both editions have consistently demonstrated the value of these advanced testing techniques in the domain of autonomous systems. The competition has thus not only fostered innovation but also served as a powerful, community-driven benchmark that reinforces the importance of the research direction explored in this dissertation.

Finding 3 (Community Benchmark): The *Aerialist* and *Surrealist* frameworks provided a successful and trusted foundation for the international UAV Testing Competition. By serving as the core benchmarking platform, they have enabled and stimulated further research, attracting and evaluating numerous advanced testing techniques from the community.

4.6 Chapter Summary

This chapter introduced **Surrealist**, a generic, two-phase methodology for bridging the reality gap in simulation-based testing by generating realistic and challenging test cases for autonomous robotic systems. The approach first achieves realism by replicating behaviors from real-world operational logs in a simulation environment. It then generates novel, challenging scenarios by systematically searching in the neighborhood of this replicated reality to find safety-critical edge cases.

The effectiveness of this methodology was rigorously validated through a case study on autonomous UAV navigation. The experiments demonstrated that Surrealist can faithfully replicate real-world flight trajectories with high precision and, subsequently, discover latent bugs in the UAV's navigation logic by generating scenarios that lead to unsafe behaviors and crashes. The framework is open-source and publicly available on GitHub[3]. The value of this approach was further validated by the establishment of the international UAV Testing Competition, which uses the frameworks presented in this thesis as its core benchmarking platform, providing community proof of the usefulness and impact of the proposed test generation infrastructure.

This chapter has shown that Surrealist can achieve faithful flight replication and subsequently manipulate the environment to find challenging conditions that lead to

[3] https://github.com/skhatiri/surrealist

unsafe behavior, establishing it as a powerful tool for quality assurance. The next chapters of this dissertation will build upon this foundation, first by directly addressing the external validity of Surrealist through its application to a completely different robotic domain in an industrial setting and then by exploring runtime safety monitoring approaches taking advantage of the vast simulation datasets generated by Surrealist.

Chapter 5

Industrial Adoption of the Proposed Simulation-based Testing Framework

The preceding chapters have introduced a general-purpose framework for addressing the reality gap in simulation-based testing. The *Aerialist* framework provides the foundational infrastructure for automated, scalable, and repeatable test execution, while the *Surrealist* methodology offers a novel, two-phase approach for generating test cases that are both realistic and challenging. While the effectiveness of this framework was rigorously validated on Unmanned Aerial Vehicles (UAVs), a critical question for any academic research remains: can the proposed solution be generalized to new domains and successfully adopted in a real-world, industrial context to deliver tangible value?

This chapter directly addresses this question by presenting a comprehensive case study on the industrial adoption and evolution of the testing framework. Through a close, one-year collaboration with **ANYbotics**[1], a world leader in industrial quadrupedal robotics, the framework was not only adapted and deployed but also extended in a novel direction that significantly accelerated their development workflow [55].

This effort serves two primary purposes for this dissertation. First, it acts as a crucial **cross-domain validation**, demonstrating that the core principles of Aerialist and Surrealist can be successfully applied from aerial robotics to the distinct challenges of legged, ground-based robots. This was achieved by integrating the framework with ANYbotics' high-fidelity Gazebo simulator for rigorous, system-level testing.

Second, it provides a definitive demonstration of **industrial adoption**. Integrating the framework with Gazebo was highly successful, equipping the ANYbotics team with a powerful tool for rigorous system-level testing. This capability enabled them to effectively find failures and objectively benchmark different navigation algorithms [55], representing a significant improvement over previous manual methods.

[1] https://www.anybotics.com/

This success motivated the natural next step of the collaboration: extending the framework's principles directly into their **MLOps pipeline**. By adapting the test execution to run in the massively parallel Isaac Gym simulator, we dramatically shortened the feedback cycle for ML engineers. This transition allowed for the rapid, automated testing and benchmarking of new navigation models immediately after training—reducing test suite execution time from over four hours to under ten minutes and enabling statistically robust metrics through multiple parallel evaluations.

This chapter details both the initial cross-domain transfer and the subsequent evolution, offering empirical insights into how the framework enhanced development efficiency, improved system robustness, and became an integral part of a modern robotics verification pipeline.

5.1 The Challenge of Ensuring Robustness in Industrial Robotics

Mobile autonomous robots are revolutionizing industrial workflows in sectors like manufacturing, logistics, and inspection by improving efficiency, productivity, and safety [36, 47]. The ANYmal D quadrupedal robot, shown in Figure 5.1, exemplifies this trend. Deployed in complex and dynamic environments like industrial plants, it handles inspection tasks that are often hazardous or impractical for humans [32, 42]. This creates an urgent demand for robust and reliable platforms that can be safely operated in the real world [33, 58].

For industrial robots such as ANYmal, autonomous navigation is a vital capability [83]. The ability to plan and follow paths while safely avoiding obstacles is paramount. As the environment in Figure 5.1 illustrates, a failure in navigation can lead to mission-critical consequences, including collisions, equipment damage, operational downtime, and significant safety risks [24]. Consequently, rigorous and comprehensive testing is not just a quality assurance step but a fundamental requirement for industrial deployment [17, 58, 59].

However, traditional testing of robotic navigation systems, which often relies on manual test design and physical trials, faces significant limitations [2, 4, 8]. This approach is notoriously costly, time-consuming, and fails to cover the full spectrum of diverse and unpredictable scenarios that a robot might encounter in a real industrial facility [16, 58, 122]. While physical testing is an indispensable final step, it is particularly risky and impractical for exploring the edge cases where subtle but critical failures often hide.

It is precisely this gap—the inefficiency and risk of manual hardware trials—that makes simulation-based testing such a flexible and necessary alternative. Simulation

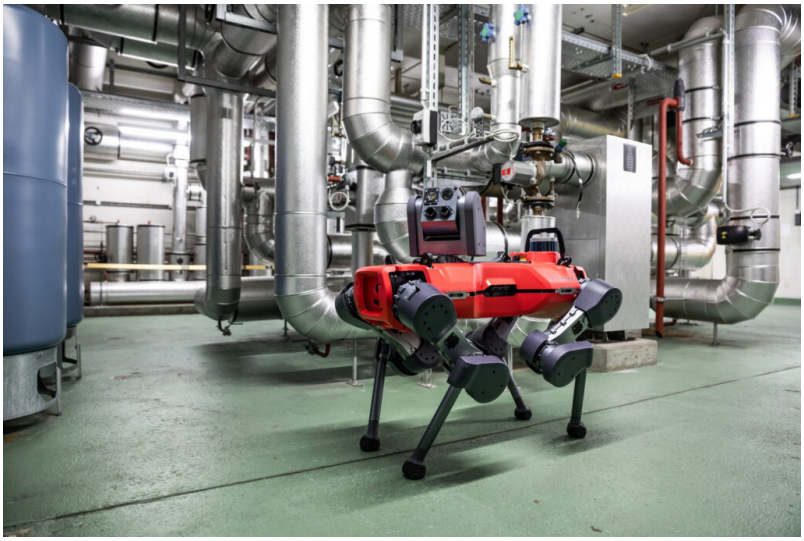

Figure 5.1. ANYmal performs inspection tasks in a typical, cluttered industrial environment. Reliable navigation is critical to its mission success and safety [7].

enables the safe, repeatable, and cost-effective testing of navigation and control logic in high-fidelity virtual environments [3, 108]. As established in Chapter 4, the primary challenge in this domain is the creation of test scenarios that are both realistic and challenging enough to be meaningful.

This chapter investigates the industrial adoption of the *Surrealist* framework, which was designed to directly target this challenge. This industrial case study was the result of a deliberate and careful process of technology transfer, beginning with a targeted search for a partner whose technical needs aligned with the solutions presented in this thesis. After identifying ANYbotics as a world leader in a field with analogous navigation challenges, we entered a collaborative exploration phase lasting several months. This period involved not only deep-dive technical discussions to identify a mutually interesting and relevant topic for both parties, but also the establishment of a formal legal framework. By signing a Non-Disclosure Agreement (NDA) and a clear Intellectual Property (IP) agreement between the university and the company, we built the necessary trust to protect ANYbotics' proprietary software stack while enabling open research collaboration. This foundational legal and technical alignment allowed us to define a clear integration plan, moving the framework from an academic prototype to an industrial-ready solution.

The collaboration was thus initiated with ANYbotics, using their flagship platform, the ANYmal D quadrupedal robot introduced in Section 3.2.2. The integration of the testing framework consisted of an initial pilot phase followed by a full deployment.

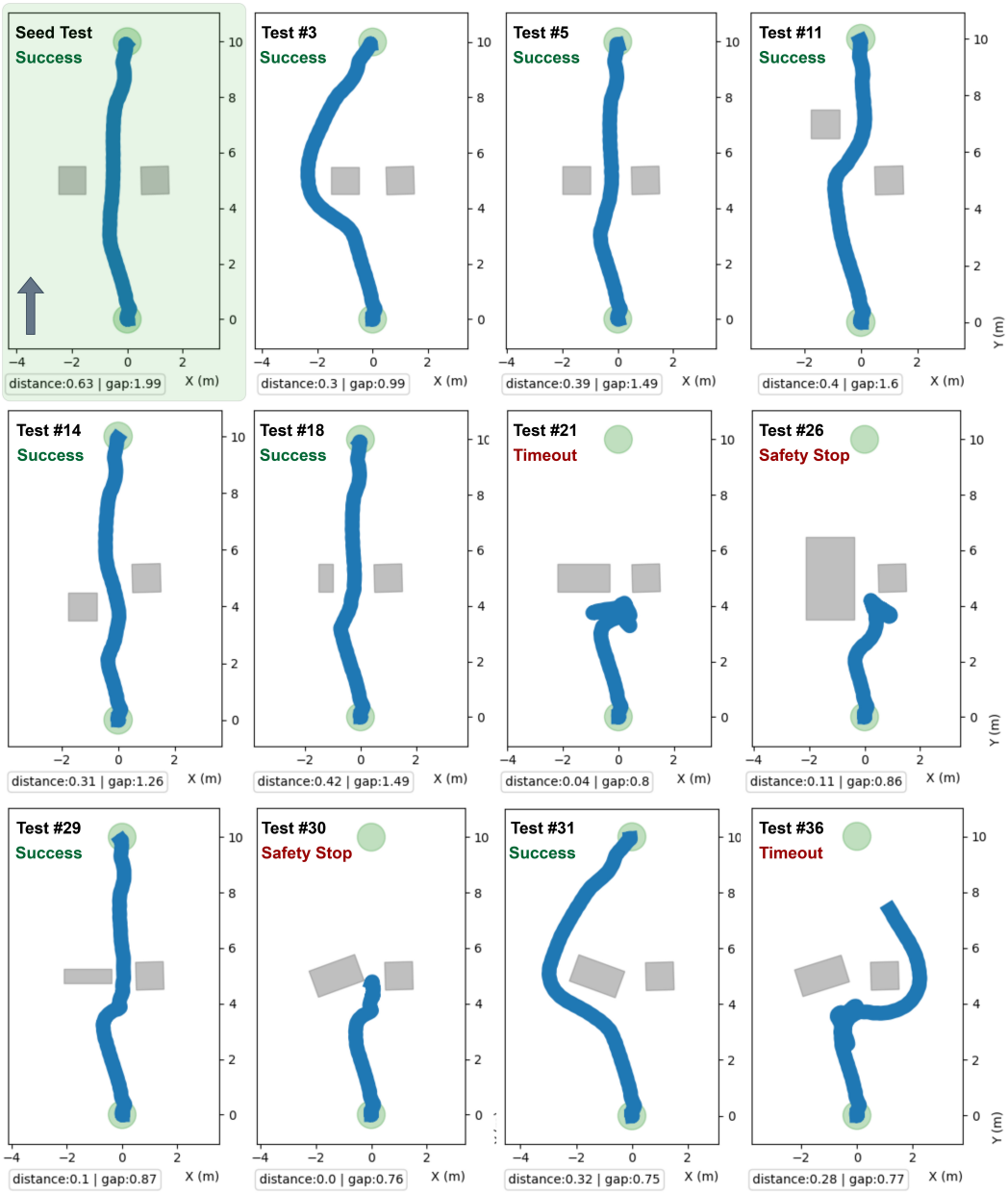

Figure 5.2. The Surrealist test generation process for a 2D robot navigation task. Starting from a manually defined seed test, the system iteratively modifies the environment by moving, resizing, and rotating obstacles to systematically search for challenging scenarios that induce navigation failures. This automated exploration uncovers critical failure cases that are often missed by manual or purely randomized testing [55].

The initial pilot study, using two experimental navigation algorithms, proved highly effective. For instance, automatically generated test suites targeting difficult obstacle avoidance scenarios (as illustrated in Figure 5.2) revealed critical weaknesses in one algorithm, leading to a mission success rate of just **40.3%**. The same test suite then served as an effective benchmark to demonstrate the superior robustness of an improved algorithm, which achieved a **71.2%** success rate.

Following this successful pilot, the ANYbotics team integrated the framework into their development workflow for a six-month industrial evaluation. They used it to assess their current released navigation stack and to benchmark four internal candidates for the next product version. A formal survey completed by eight ANYbotics engineers confirmed the framework's value, with participants reporting that it significantly streamlined their workflow, proved effective at uncovering algorithm deficiencies missed by manual methods, and enhanced their ability to objectively benchmark improvements. This chapter details the methodology and results of this industrial case study.

5.2 Adapting the Testing Framework for a New Robotic Domain

A key test of the framework's generalizability was its application to a new and distinct robotic domain: quadrupedal ground robots. This transition required adapting the tools and methodologies originally designed for UAVs to meet the specific challenges of legged locomotion and navigation. This section details the technical approach for this cross-domain transfer, highlighting the modifications made to the system under test interface, the Aerialist test bench, and the Surrealist test generator. The overall integration architecture is depicted in Figure 5.3.

5.2.1 System Under Test Adaptations

The ANYmal robot possesses a large, complex codebase and a sophisticated, proprietary software architecture. To integrate the research prototypes (Aerialist and Surrealist) without intrusive modifications to this industrial-grade system, a non-invasive interface was essential. To this end, a **System Test Interface** was developed. This component acts as a facade, abstracting the complexities of the ANYmal's software setup, simulation configuration, and test execution.

Building upon ANYbotics' existing Gazebo-based simulation infrastructure, a command-line interface (CLI) was added to allow for external, automated control over the test scenarios. This interface is driven by the standardized Aerialist test definition file (as

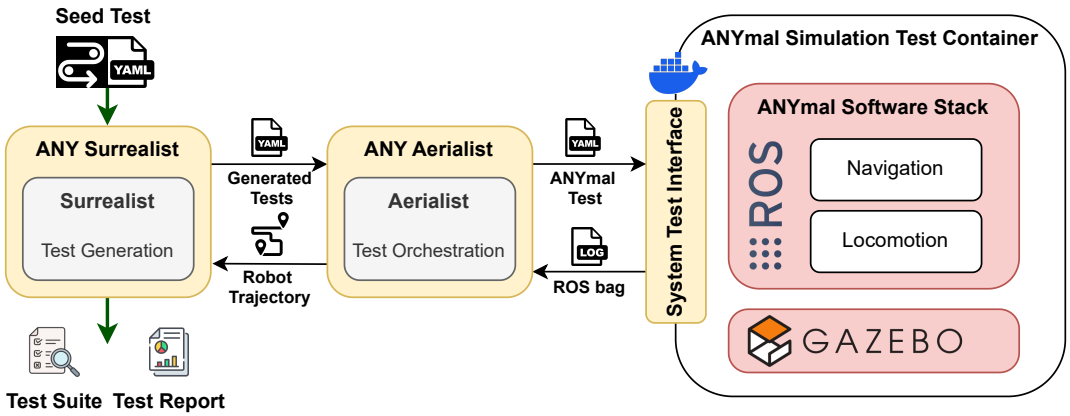

Figure 5.3. The integrated architecture for testing the ANYmal robot. The adapted ANY Surrealist and ANY Aerialist frameworks interact with the proprietary ANYmal software stack through a dedicated System Test Interface [55].

described in Section 3.3.1 and shown in Figure 5.4). This allows external tools like Surrealist to precisely specify all aspects of a test—including obstacle parameters, the robot's mission plan, and specific configurations—without modifying the internal ANYmal code. This approach was crucial for enabling a black-box testing methodology and ensuring a smooth integration into the company's existing workflows. For portability, the entire system, including the simulator, the ANYmal software, and the new interface, was dockerized.

5.2.2 Aerialist Adaptations (ANY Aerialist)

The core design philosophy of Aerialist proved vital for this cross-domain transfer. Although originally developed for the PX4-based UAV ecosystem, its modular architecture was intentionally designed to be extensible. To adapt it for the ANYmal robot, the framework was refactored, evolving it from a UAV-specific tool into a more generic test bench for robotic navigation systems. The key modifications included:

- **Refactoring for Extensibility:** The core codebase was refactored to introduce abstract interfaces for robot- and simulator-specific functions, decoupling the generic test orchestration logic from the platform-specific execution details.

- **ROS Bag Support:** The Analyst module was modified to parse **ROS bag files**, the standard data logging format in the ROS ecosystem used by ANYmal, instead of the PX4 ulog files used in the UAV case study.

- **New Failure Categories:** The automated outcome classification was extended with new categories relevant to the ANYmal's behavior, including *Success*, *Time-out*, and *Safety-Stop*—a specific failure mode where the robot's onboard safety logic halts its movement to prevent an imminent collision.
- **Updated Geometric Calculations:** The internal models were updated to treat the ANYmal as a box, reflecting its true physical dimensions for more accurate distance-to-obstacle calculations, rather than as a point-mass, which was sufficient for the UAV.
- **Enhanced Plotting:** The automated visualizations were enhanced based on feedback from ANYbotics engineers to include thicker trajectory lines representing the robot's physical footprint and annotations for the minimum robot-to-obstacle distance and obstacle-to-obstacle gap. As shown in Figure 5.4, these improved plots were a key tool for the engineering team, enabling them to quickly diagnose test failures.

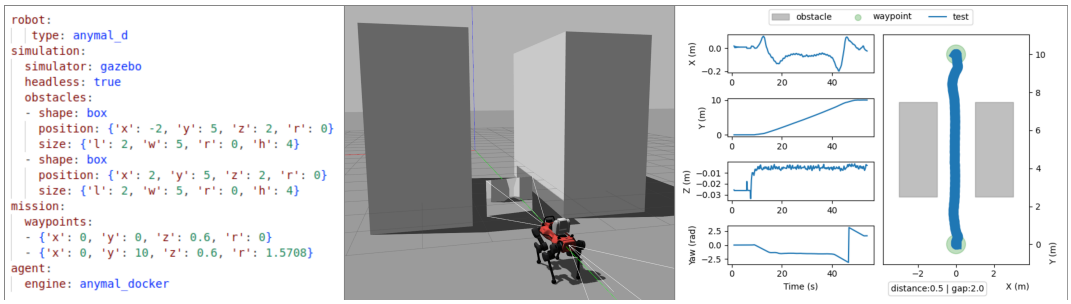

Figure 5.4. A sample test case for the ANYmal robot, as managed by the adapted Aerialist framework. The figure shows the YAML test description (left), the corresponding 3D simulation environment in Gazebo (middle), and the automated trajectory visualization generated by the Analyst module (right) [55].

5.2.3 Surrealist Adaptations (ANY Surrealist)

Because Surrealist's dependencies on the underlying robotic platform were already abstracted by Aerialist, its core search-based logic could be reused for the ANYmal with minimal changes. However, to optimize its effectiveness for quadrupedal navigation, several enhancements were made based on direct feedback from the ANYbotics team:

- **Adapted Fitness Function:** The fitness function for challenge generation was kept conceptually the same—aiming to minimize the robot's distance to obstacles—

as ANYbotics engineers confirmed this remained a good proxy for generating challenging and failure-inducing scenarios.

- **Industry-Relevant Seed Scenarios:** New seed tests were manually crafted to reflect common and difficult scenarios in industrial inspection tasks, such as navigating through narrow corridors and standard-sized doorways.
- **Waypoint Mutation:** A new mutation operator was implemented to allow the search to manipulate the robot's mission waypoints (e.g., its starting position) in addition to the obstacles. This was a specific request from the engineers to find corner cases related to how the robot perceives obstacles from different angles.
- **Support for Regression Testing:** A feature was added to easily re-execute an entire test suite, which proved essential for enabling regression testing across different versions of the navigation algorithms.
- **Extended Metrics:** Logging and analysis were extended to capture quadruped-specific metrics, including path length and deviation, and to provide aggregated summaries of test outcomes and performance statistics.

5.2.4 Integration Process

Before the technical integration could begin, the primary challenge was to navigate and understand ANYbotics' large-scale, proprietary software stack and robot platform. This involved a substantial learning curve to acquire the necessary "know-how"— understanding their internal frameworks, build systems, ROS-based architecture, and specific simulation environment configurations. A significant portion of the integration process was dedicated to this knowledge transfer. Working closely with their engineers, the main obstacle was identifying the correct, non-invasive integration points within their mature code base and determining how to adapt the framework to respect their existing workflows.

Once this deep understanding was established, the technical adaptation of the framework proceeded. To facilitate the integration, the public *Surrealist*[2] and *Aerialist*[3] repositories were forked to create ANYmal-specific versions. The core frameworks were refactored to enhance modularity, while ensuring the forks remained synchronized with the main repositories for straightforward maintenance. This design proved highly effective, as the adaptation required minimal new code: *Surrealist* was extended with just three new classes inheriting from generic base classes (∼350 LOC), and *Aerialist*

[2] http://github.com/skhatiri/Surrealist
[3] http://github.com/skhatiri/Aerialist

with two (∼250 LOC). These minimal additions highlight the framework's modularity and provide a clear blueprint for extending it to other robotic use cases, following the detailed guidelines documented in the public repositories.

The full integration process, including the development of the interface and the adaptation of the frameworks, was completed over a three-month period, facilitated by bi-weekly meetings and continuous feedback from the ANYbotics team. This collaborative and iterative approach was essential for successfully adapting the research framework to meet the rigorous demands of an industrial application.

5.3 A Two-Phase Industrial Evaluation Methodology

To empirically evaluate the adapted framework's ability to enhance quadrupedal robot navigation robustness, a comprehensive, two-phase study was conducted in collaboration with ANYbotics. The evaluation combines quantitative metrics from an initial pilot study with rich qualitative feedback from a long-term industrial deployment. The study was guided by the following four research questions:

- **RQ_1 (Development Process):** *How does the integrated framework improve the development workflow and testing practices at ANYbotics?*
- **RQ_2 (Failure Detection):** *How effective is Surrealist at generating test cases that reveal failures in ANYmal's obstacle avoidance?*
- **RQ_3 (Improvement Assessment):** C*an Surrealist track and quantify performance improvement across navigation software versions?*
- **RQ_4 (System Verification):** *How does the integrated framework enhance the verification process for the ANYmal navigation, ensuring robustness and reliability?*

The first phase of the study, a pilot, focused on an initial validation of the framework on two experimental navigation algorithms to gather quantitative data. The second phase involved integrating the framework into ANYbotics' daily workflow for a six-month period, providing qualitative and empirical evidence of its industrial impact and value.

5.3.1 Phase 1: Pilot Study

The first phase was a three-month pilot study designed to provide an initial, controlled validation of the integrated framework on ANYbotics' navigation software. To demonstrate the approach's black-box capabilities without requiring access to proprietary

code, the evaluation used two experimental obstacle avoidance algorithms provided by the company: *Exp-Nav-A* and *Exp-Nav-B*. The pilot study involved four key steps:

1. **Scenario Definition:** Five seed test scenarios were manually crafted based on common and challenging real-world industrial use cases, such as navigating through narrow corridors or cluttered spaces.

2. **Initial Test Generation:** The Surrealist framework was used to automatically generate a test suite (TS-Exp-A) from these seed scenarios, specifically targeting the first algorithm, *Exp-Nav-A*, to analyze its performance and expose potential weaknesses.

3. **Algorithm Comparison:** The same test suite (TS-Exp-A) was then executed on a newer, improved algorithm, *Exp-Nav-B*, to provide a direct, quantitative comparison of their performance and robustness.

4. **Adaptive Test Generation:** To assess how the test generation process adapts to a more robust system, a new test suite (TS-Exp-B) was generated, this time tailored specifically to find the unique corner cases and failure modes of *Exp-Nav-B*.

Throughout the pilot, an iterative feedback loop was maintained between the author (acting as lead developer) and three supervisors from the ANYbotics navigation and testing teams. The performance of each algorithm was evaluated using metrics such as the Mission Success Rate and Safety-Stop Rate.

5.3.2 Phase 2: Industrial Deployment and Qualitative Evaluation

Following the successful pilot, the framework was deployed and integrated into the daily development workflow of the ANYbotics navigation team for a six-month period. A senior engineer, who was not involved in the pilot, led the use of the tool for failure identification, performance analysis, and to guide improvements for the company's proprietary navigation stack. During this phase, the framework was used to test multiple software versions, including the current release, referred to as *ANY-Nav-A*, and four internal release candidates, referred to as *ANY-Nav-*B_{1-4}. The main activities included:

1. **Benchmark Generation:** The ANYbotics team used Surrealist to generate a benchmark test suite (TS-ANY-A) for the current release, *ANY-Nav-A*. This suite was used to assess its baseline performance and identify core deficiencies to be addressed in the next version.

2. **Regression Testing and Comparison:** The benchmark suite (TS-ANY-A) was subsequently used for regression testing throughout the development of the next release. The four candidate algorithms (*ANY-Nav-*B_{1-4}) were executed against this suite to quantitatively assess performance improvements and behavioral differences.

3. **Targeted Failure Replication:** After two new types of failures were discovered during physical field tests of *ANY-Nav-A*, Surrealist was used to perform targeted test generation. The goal was to create simulation-based test suites that could reliably reproduce these specific failures, enabling efficient diagnosis and resolution by the development team.

Due to the confidential nature of the quantitative results from the six-month deployment (including the performance of the proprietary algorithms and the specific failures found), the framework's industrial impact was primarily evaluated through a formal, qualitative survey. To ensure all participants had a shared context, engineers from various teams (navigation, locomotion, perception, and verification) first watched a 14-minute video demonstrating the framework's workflow. The survey then collected data on the usability of the tools, the effectiveness of the generated tests for failure detection and benchmarking, and the framework's overall impact on their development process and confidence in the system. The findings from this evaluation are presented in the next section.

5.4 Evaluation Results and Industrial Impact

The two-phase evaluation provided both quantitative data from the pilot study and rich qualitative insights from the long-term industrial deployment. This section presents these findings, organized by the framework's observed impact on the development workflow (RQ_1), its effectiveness at detecting failures (RQ_2), its utility for benchmarking (RQ_3), and its overall contribution to system verification (RQ_4). The evaluation involved a formal survey with eight experienced engineers from diverse teams at ANYbotics, including navigation, locomotion, perception, and verification. All participants held advanced degrees (Master's or PhD), and seven of the eight had more than three years of professional robotics experience.

5.4.1 Streamlined Development Workflow (RQ_1)

A primary goal of the framework was to improve the efficiency and usability of the testing process. Feedback from both the pilot and deployment phases confirmed a significant positive impact on the development workflow.

Table 5.1. Performance comparison of the pilot test subjects.

Subject	Test Suite (#)	$boxes_1$		$boxes_2$		corridor		cylinders		L-corridor		overall	
		Succ.	Stop	Succ.	Stop	Succ.	Stop	Succ.	Stop	Succ.	Stop	Succ.	Stop
Exp-Nav-A	TS-Exp-A (429)	39.2%	40.5%	42.1%	44.9%	37.5%	39.8%	75.0%	23.2%	23.1%	52.9%	40.3%	42.2%
Exp-Nav-B	TS-Exp-A (429)	81.1%	5.4%	69.2%	0.0%	75.0%	12.5%	98.3%	1.8%	48.1%	16.4%	71.2%	7.7%
Exp-Nav-B	TS-Exp-B (422)	72.5%	12.2%	89.8%	8.3%	64.7%	11.8%	76.3%	22.0%	34.8%	5.6%	68.3%	11.1%

During the pilot phase, the iterative process of gathering feedback from ANYbotics engineers was crucial for refining the framework's usability. Many key features, such as the improved plotting in Aerialist and the waypoint mutation capability in Surrealist, were direct results of this close collaboration.

In the subsequent deployment phase, a formal survey confirmed the framework's high usability and positive effect on the engineers' workflow. The YAML-based test definition in Aerialist was rated highly for its ease of use, with an average score of 4.5 out of 5; notably, **75% of participants (6 out of 8) found it "easier" or "much easier" to use than their previous approaches**. The built-in automated visualizations were also praised, receiving an average rating of 4.3, with 60% of participants considering them "better" or "much better" than alternative tools. Regarding the automated test generation with Surrealist, participants found it highly intuitive to generate a test suite from a seed scenario (average rating of 4.1). In terms of efficiency, while 50% considered it comparable to existing methods, they noted the main benefit was the automation, and 37% reported a "significant improvement" in overall speed.

Finding 4 (Streamlined Workflow): The integrated framework streamlined industrial testing workflows. The Aerialist component was rated highly for usability and visualization, with 75% of engineers preferring it over prior tools. The Surrealist component was praised for its intuitive test generation, leading to reduced manual effort and a more productive test creation process.

5.4.2 Effective Failure Detection (RQ_2)

The evaluation demonstrated the framework's powerful capability for automatically generating test scenarios that reveal critical, previously unknown failures.

The quantitative results from the pilot study provided strong initial evidence. The test suite *TS-Exp-A*, which was automatically generated by Surrealist, proved highly effective at exposing weaknesses in the experimental *Exp-Nav-A* algorithm. As shown in Table 5.1, the algorithm achieved a low overall success rate of just **40.3%**, with safety-stops accounting for 42.2% of outcomes. The generated scenarios were particularly adept at finding challenging corner cases; for instance, in the *L-corridor* scenario, the

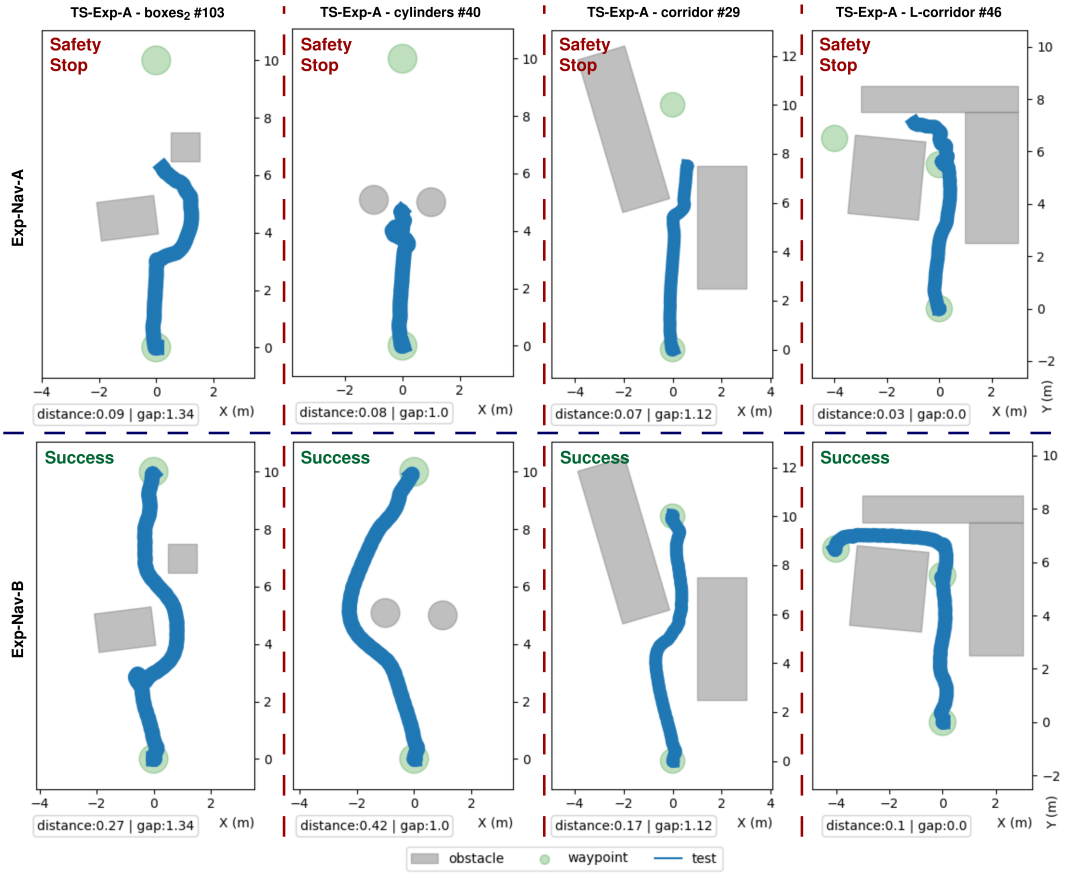

Figure 5.5. Robot trajectory comparison with different obstacle avoidance algorithms in the same test environments [55].

success rate dropped to a mere 23.1%, highlighting a specific weakness in navigating tight spaces. These weaknesses are visually evident in the robot trajectories plotted in Figure 5.5 (top row) where the robot fails to find safe path to navigate and the fail safe mechanism is frequently triggered to avoid potential collisions to obstacles.

This quantitative finding was strongly corroborated by the qualitative feedback from the long-term deployment. All eight participants rated the framework as either **"Effective" (62%) or "Very Effective" (38%) at identifying previously unknown failures and corner cases**. One engineer's comment captured the sentiment: "Unknown edge cases. This is where surrealist shines the most!". The generated scenarios were perceived as both realistic (average rating of 4.3) and challenging (3.6), with 50% of participants finding them more challenging than their typical manually designed tests. The engineers also confirmed the relevance of the discovered failures, with one stating, "I never saw a failure case that we did not yet see with Surrealist," indicating a high

degree of fidelity between the simulated failures and real-world behaviors.

Finding 5 (Failure Detection): Surrealist's search-based approach effectively generated challenging scenarios that revealed critical failures. In the pilot study, it reduced a preliminary navigation algorithm's success rate to 40.3%. During the industrial deployment, all engineers rated it as effective for uncovering previously unknown failures and corner cases.

5.4.3 Objective Benchmarking and Improvement Assessment (RQ_3)

A key capability of the framework is its support for re-executing test suites, which provides a repeatable, quantitative method for comparing the performance of different algorithm versions.

The pilot study clearly demonstrated this value. As shown in Table 5.1 and Figure 5.6, when the improved *Exp-Nav-B* algorithm was run on the same challenging test suite (*TS-Exp-A*), it dramatically outperformed its predecessor. Its overall success rate was **71.2%** compared to 40.3%, and its safety-stop rate was only **7.7%** compared to 42.2%. Visual inspection of the trajectories in Figure 5.5 confirmed this quantitative improvement, showing that *Exp-Nav-B* consistently executed smoother, more confident paths. Furthermore, the evaluation showed that the framework could adapt to this improved robustness. When a new test suite (*TS-Exp-B*) was generated specifically for *Exp-Nav-B*, its success rate dropped from 71.2% to 68.3%, demonstrating that Surrealist could find new, more difficult corner cases tailored to the weaknesses of the improved algorithm.

The value of this objective benchmarking capability was strongly confirmed during the industrial deployment. The ANYbotics team actively used the test suite re-execution feature to benchmark new algorithm candidates against the current release and to inform their enhancement decisions. In the survey, all participants agreed that the framework was useful for comparing different algorithms, rating its effectiveness higher than their previous manual methods with an average score of **4.6 out of 5**.

Finding 6 (Benchmarking): The framework enables repeatable, quantitative comparisons of navigation algorithms. The pilot study clearly distinguished the performance of two algorithms (71.2% vs. 40.3% success rate). During deployment, engineers rated this benchmarking feature highly (4.6/5) and used it to guide their development efforts.

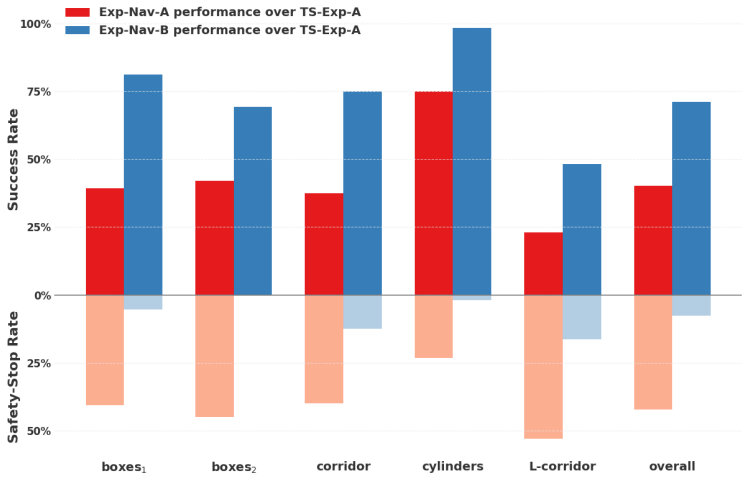

Figure 5.6. Performance comparison of the pilot test subjects for test suite TS-Exp-A. The second algorithm (Exp-Nav-B, blue) showed a clear and significant improvement over the initial prototype (Exp-Nav-A, red), with a much higher success rate and a lower rate of safety-stops across all scenarios [55].

5.4.4 Enhanced System Verification and Confidence (RQ_4)

Ultimately, the integration of the framework had a significant positive impact on the overall verification and validation (V&V) process at ANYbotics, increasing both the rigor of the testing and the engineers' confidence in the final product.

The survey results from the deployment phase were unequivocal on this point. **All participants "agreed" or "strongly agreed" that the framework increased their confidence in the system's robustness**, and a strong majority (7 out of 8) stated that it improved their team's overall verification ability. This was attributed to its effectiveness in specific V&V tasks. For finding difficult-to-predict corner cases, all participants rated the system as "better" or "much better" than manual design. The ability to re-run test suites for regression testing was considered "extremely valuable" by 86% of the engineers. The generated scenarios and visualizations were also found to be very helpful in debugging the root causes of failures (average rating of 3.9).

The framework's impact was felt across the broader development workflow. A majority of participants (71%) found the approach "very valuable" for prioritizing physical, real-world tests, allowing them to focus costly hardware testing on the most critical scenarios identified in simulation. The framework's critical role in their process was best summarized by one engineer who stated they "would not release [new navigation algorithms] if they were not tested with surrealist," positioning the tool as an **essential pre-release validation gate**.

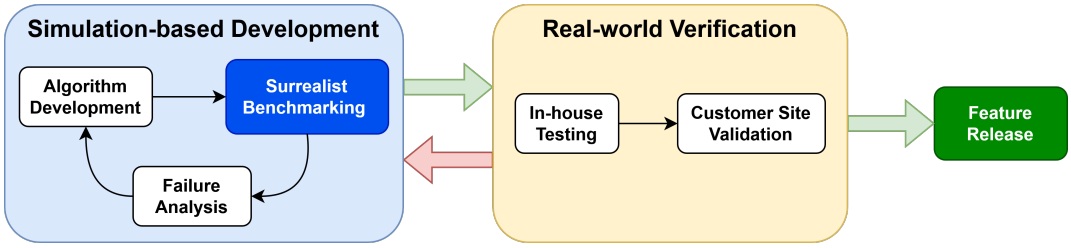

Figure 5.7. ANYbotics' development workflow for releasing new navigation algorithms includes both simulation and real-robot testing stages. Our approach integrates directly into the simulation stage, allowing for early detection of critical failures and deficiencies before they reach real-robot tests [55].

Finding 7 (Verification): The framework enhances system verification and developer confidence by systematically uncovering difficult-to-predict corner cases and enabling efficient regression testing. Participants rated it superior to manual test design, found its debugging support highly useful, and adopted it as an essential pre-release validation step.

5.5 From System Validation to Early-Stage Benchmarking: An MLOps Extension

The successful deployment of the search-based testing framework in Gazebo demonstrated its significant value for system-level validation [55]. However, the ANYbotics development workflow for navigation capabilities, as illustrated in Figure 5.7, highlighted an opportunity for even greater impact. Their process involves two main simulation stages: first, ML models for navigation are trained and evaluated in NVIDIA's Isaac Gym, a high-performance simulator capable of massive parallelism. Second, the most promising models are integrated into the full, complex software stack for more rigorous, high-fidelity testing in Gazebo.

While the Gazebo-based integration of Surrealist provided powerful tools for system-level validation, its execution time (over four hours for a 400-test suite) created a bottleneck, delaying feedback for ML engineers. This inspired a powerful extension of the framework: adapting the Surrealist methodology to operate directly in the initial, high-performance training environment. By "shifting left" and integrating into the MLOps pipeline, this extension provides immediate, large-scale testing capabilities, dramatically accelerating the development and benchmarking of new ML models.

5.5.1 A Massively Parallel Simulation Environment with Isaac Gym

To achieve this performance leap, the testing environment was re-engineered to fully leverage the capabilities of a GPU-accelerated simulator. This work utilized **NVIDIA Isaac Gym**, a high-performance physics simulation environment designed for robotics research, particularly for training reinforcement learning (RL) models where massive data throughput is essential [76]. Unlike traditional simulators that run on the CPU, Isaac Gym performs all physics calculations directly on the GPU, enabling the parallel simulation of thousands of robots in real-time.

To effectively leverage the computational potential of this environment, a simple black-box integration was insufficient. Instead, the core, valuable features of the *Aerialist* framework were deeply integrated into ANYbotics' proprietary ML training infrastructure. The *Aerialist* YAML test definition model was adopted as the standard input format. A new interface was developed to consume entire test suites (folders of YAML files) and programmatically translate them into the "train" of test environments seen in Figure 5.8. This setup allows for the concurrent evaluation of hundreds of ANYmal agents, each tackling a different scenario or a repeated run of the same scenario.

This approach yielded a dramatic >25x speedup, reducing the execution time for a 400-scenario test suite from over four hours to less than ten minutes on a single development laptop. With this high-throughput evaluation, the focus shifted from analyzing detailed, individual logs (like ROS bags) to aggregating high-level performance statistics. While *Surrealist* can still be used to generate new test suites for Isaac Gym, the primary industrial use case for this extension is the rapid benchmarking of newly trained models against large, existing test suites.

5.5.2 Automated Metrics for MLOps Integration

The high-throughput nature of the Isaac Gym environment enables the rapid collection of large-scale, statistically significant performance data. This capability was integrated directly into the MLOps pipeline at ANYbotics, providing ML engineers with immediate feedback. The framework automatically computes and logs a hierarchical set of metrics to a TensorBoard dashboard for real-time analysis and historical tracking.

Per-Test-Case Metrics

For each test case, executed with multiple parallel agents, the framework computes metrics to assess both general performance and non-deterministic ("flaky") behavior:

- **Rate-Based Metrics:** These quantify the average performance across all agents in a single scenario.

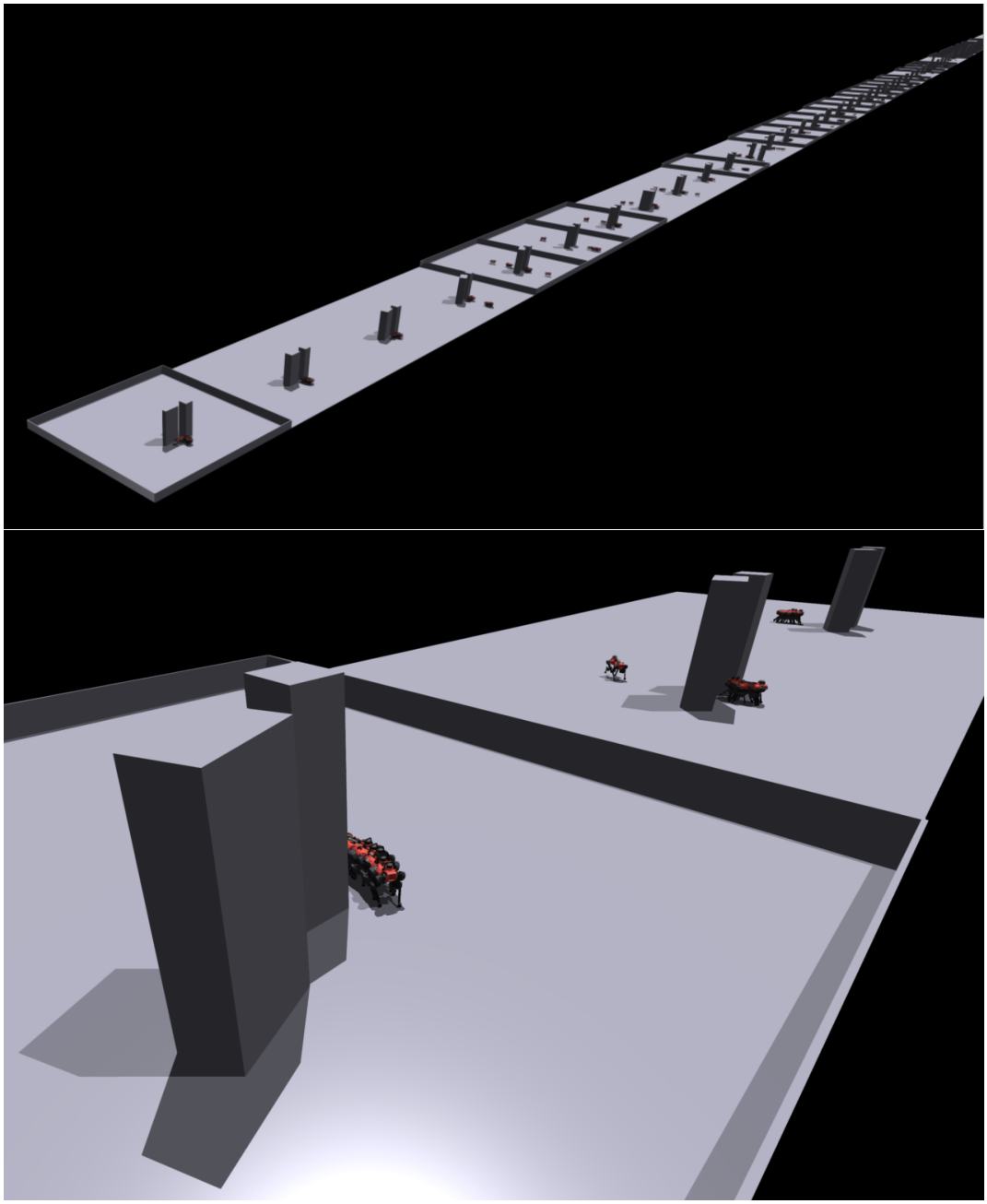

Figure 5.8. The massively parallel testing environment in Isaac Gym. Up: An entire test suite is arranged as a sequential "train" of environments. Down: Multiple ANYmal agents are spawned to execute tests concurrently, enabling rapid and statistically significant evaluation.

  - *Success Rate:* The percentage of agents that successfully navigate from start to goal.
  - *Collision Rate:* The percentage of agents that collide with an obstacle.
  - *Timeout Rate:* The percentage of agents that fail to reach their goal within the time limit.

- **Flakiness Detection Metrics:** These boolean metrics identify if a failure of any type occurred at least once, which is crucial for highlighting scenarios that trigger rare or non-deterministic bugs.
  - *Had Success:* True if at least one agent successfully navigated during any of the runs.
  - *Had Collision:* True if at least one agent collided during any of the runs.
  - *Had Timeout:* True if at least one agent timed out.

Aggregate Test-Suite Metrics

To provide a high-level benchmark of a model's overall capabilities, the per-test-case metrics are aggregated across the entire test suite (or specific subset of scenarios, e.g., the narrow corridors) into two distinct scores:

- **Average Performance Score:** This score reflects the model's general robustness and is calculated by averaging the rate-based metrics across all test cases (e.g., *Average Success Rate*, *Average Collision Rate*, *Average Timeout Rate*).
- **Failure Discovery Score:** To ensure that rare but critical failures are not diluted by averaging, this score is computed from the flakiness metrics. For example, the *Any-Collision Rate* and *Any-Timeout Rate* represent the percentage of test cases in the suite that respectively registered at least one collision or timeout. This score effectively measures the breadth of failure modes discovered, highlighting whether a model fails in any of the challenging scenarios, even if inconsistently.

This dual-metric approach provides engineers with an immediate, comprehensive summary of a model's performance, enabling quick and objective benchmarking. This tight feedback loop is invaluable for identifying performance improvements and, crucially, for detecting regressions as soon as they are introduced. This extension transforms the Surrealist methodology from a powerful system-level validation tool into a rapid, early-stage evaluation framework, improving the quality of navigation models long before they are integrated into the full robotic system for final, high-fidelity testing.

5.6 Industrial Impact, Lessons Learned, and Limitations

5.6.1 Industrial Impact and Adoption

The research presented in this chapter demonstrates the successful adaptation and industrial integration of the *Surrealist* and *Aerialist* frameworks into the development workflow at ANYbotics. The successful deployment for testing a proprietary navigation stack marks a key milestone, transitioning the search-based testing methodology from a research prototype to a solution demonstrated in its intended operational environment. This achievement corresponds to a Technology Readiness Level (TRL) 7 [77], a level of maturity that is rarely reached by academic research projects.

The framework's adoption and impact are clearly reflected in the survey results: half of the participants identified themselves as active users, and 75% rated its impact on their team's workflow as "very positive". The majority reported they would continue to use the tool on a monthly or weekly basis, a strong indication of its sustainable integration into their development processes. This trust was built on the framework's ability to generate meaningful and relevant test cases. As one engineer noted, the tool's capability to create *"test cases that reflect challenges observed later in the real world [...] built up trust quite a bit."* This success has also inspired broader adoption within the company, with the locomotion team now exploring a similar search-based approach, demonstrating an impact beyond the initial navigation context.

5.6.2 Takeaways for Researchers and Practitioners

The close, year-long collaboration provided several key insights into bridging the gap between academic research and industrial practice in robotics.

- **Integrate with a Non-Invasive Abstraction Layer:** The success of this project was critically dependent on the creation of the System Test Interface—a facade that allowed the research prototypes (Surrealist and Aerialist) to interact with ANYbotics' proprietary software stack in a black-box, non-invasive manner. This was key for industrial adoption, as it avoided disrupting existing and complex workflows. Additionally, Aerialist's generic test description (in YAML) and its decoupled architecture were instrumental in enabling the smooth extension from UAVs to quadrupeds.

- **Adopt a User-Driven, Iterative Development Process:** Beyond the core algorithms, the practical usability of a testing tool is paramount for its adoption. The pilot phase, with its close, iterative feedback loop involving ANYbotics engineers, was instrumental in refining the framework. Direct user feedback led

to meaningful and necessary improvements in visualizations, performance metrics, and essential features like test suite re-execution, ultimately enhancing the tool's utility and ensuring its successful deployment.

- **Address the Sim-to-Real Gap Strategically:** The survey revealed that the sim-to-real gap is a relative, not absolute, barrier. It was considered manageable for navigation, where high-level algorithmic flaws could be successfully identified in simulation. The engineers' primary concerns were perception-related (e.g., lack of sensor noise, oversimplified obstacles). This reinforces a practical workflow: use simulation for broad, cost-effective discovery of high-level algorithmic and logic-based flaws, and reserve targeted physical testing for scenarios where perception performance may degrade.

5.6.3 Limitations and Future Directions

While the findings support the positive impact of the integrated framework, several limitations must be considered. The results are fundamentally dependent on the fidelity of the Gazebo simulation environment and are subject to the well-known "reality gap". The test generation process was rooted in a limited set of manually designed seed scenarios and was guided by a fitness function focused primarily on obstacle proximity; these choices, while effective, may not capture all relevant edge cases or dimensions of scenario difficulty. Finally, the qualitative insights are derived from a small group of eight engineers at a single company, and the quantitative results are specific to the ANYmal robot.

The feedback from the study participants also highlighted a clear path for future evolution. Key directions include further bridging the sim-to-real gap by incorporating more complex 3D terrains and dynamic obstacles; streamlining the debugging process by introducing a "real-to-sim" capability to automatically generate test seeds from real-world failure data; and extending the framework's scope to new domains, such as locomotion testing.

On a broader note, this industrial validation serves as a strong proof of concept for the generalizability of the thesis's core methodology. The successful adaptation from aerial robots (UAVs) to a complex legged robot (ANYmal D) demonstrates that the framework's core principles—a non-invasive abstraction layer, a standardized test definition, and a black-box, search-based approach—are not tied to a single platform. This suggests that the same concepts can be transferred to other industrial robotics companies and entirely new domains. Promising future applications could include Autonomous Ground Vehicles (AGV) or humanoid robots, which, despite their different physics, often rely on a similar hierarchical control structure where a high-level navi-

gation system must be rigorously tested.

Chapter 6

Runtime Safety Monitoring of Robotic Navigation Systems

The preceding chapters have established a comprehensive framework for the pre-deployment verification of autonomous robotic systems. The *Aerialist* framework provides the infrastructure for automated test execution, while the *Surrealist* methodology enables the generation of realistic and challenging test cases that bridge the reality gap and uncover critical design flaws. However, while such rigorous, simulation-based testing significantly improves a robot's robustness, it cannot account for every possible eventuality that may arise during real-world operation. Even with the most exhaustive testing, autonomous systems may deviate from their intended behavior at runtime due to unforeseen factors such as sensor noise, unexpected environmental conditions, or complex interactions that were not covered in the test suites [58, 117, 118].

This reality necessitates a complementary layer of safety assurance: **runtime monitoring**. To ensure safe operation, especially in safety-critical applications, it is crucial to have mechanisms that can actively monitor the system's behavior during its mission and provide an early warning of impending failures. The current commercial deployment of autonomous systems legally reflects this need, often requiring a human pilot to actively monitor operations and be prepared to assume control if necessary [78]. The core challenge, therefore, is to develop automated methods that can reliably detect precursors to unsafe behavior.

This chapter proposes that observable **Decision Uncertainty**—characterized by inconsistent, erratic, or unstable patterns in a robot's high-level control signals—can serve as such a precursor. While previous work has investigated uncertainty in the context of specific machine learning components [79, 103], the relationship between a robot's holistic, observable decision-making behavior and its subsequent safety state has remained largely unexplored.

To investigate this, the chapter presents a two-part study. First, we define the concept of *decision uncertainty* and detail a large-scale empirical investigation into its correlation with safety violations in the context of autonomous UAV navigation. Using a dataset of over 5,000 simulated flights, we quantify the extent to which uncertain decisions lead to unsafe states. Second, based on the findings of this study, we introduce and evaluate **Superialist**, a lightweight, black-box runtime monitor. Superialist uses an autoencoder to detect decision uncertainty in real-time from high-level control signals, providing a practical mechanism for the early prediction of unsafe states.

6.1 An Empirical Study on Uncertainty and Safety in UAV Navigation

To establish a solid, evidence-based foundation for a runtime monitor, we first conducted a large-scale, data-driven empirical investigation into the relationship between decision uncertainty and flight safety. The primary goal was to answer a fundamental question:

- **RQ_1 (Uncertainty vs. Unsafety)**: *Is there a quantifiable correlation between the observable uncertainty in a UAV's decision-making process and the safety of its flight trajectory?*

Figure 6.1 visually illustrates the core concepts of this study. The flight trajectories show four similar test cases that result in different outcomes across two dimensions: safety (safe vs. unsafe, based on proximity to obstacles) and certainty (certain vs. uncertain, based on the smoothness and consistency of the flight path). The right-hand column (b, d) shows clear signs of decision uncertainty—hesitant, erratic movements—while the left-hand column (a, c) shows more direct, certain paths. The study aimed to systematically quantify the relationship between these observable behaviors.

6.1.1 Defining Flight Quality Attributes

Before presenting the methodology and results, it is essential to formally define the key attributes of flight quality that were investigated: decision uncertainty, flight safety, and behavioral non-determinism.

Decision Uncertainty. While the concept of uncertainty in autonomous systems is well-established, it is often tied to the internal state of a specific model (e.g., *epistemic uncertainty* in a neural network) or the inherent randomness of the environment (*aleatoric uncertainty*) [52]. In this study, we focus on a different, albeit related, aspect: the observable, behavioral effects that arise when a system encounters a situation

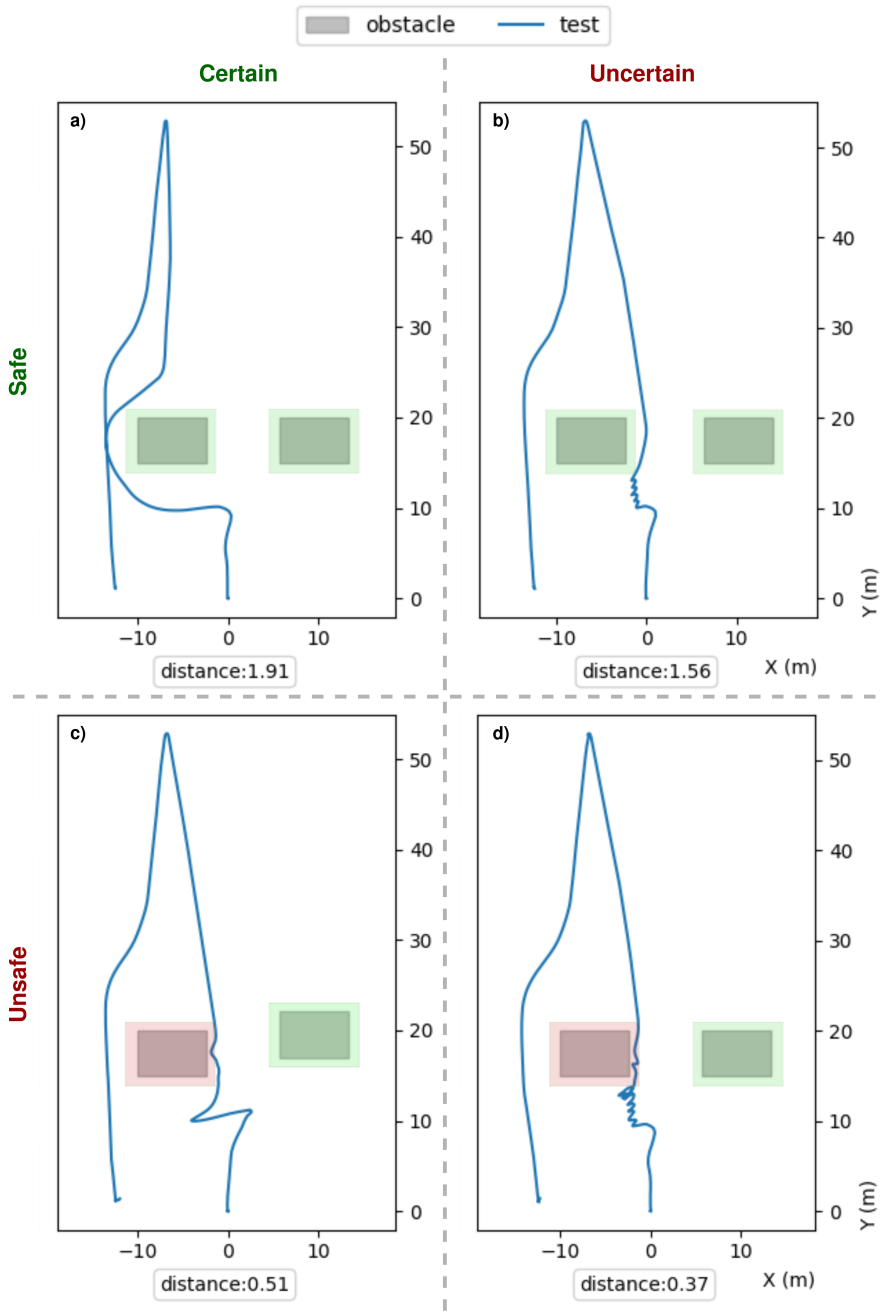

Figure 6.1. Examples of flight outcomes across the dimensions of safety and certainty. The study aimed to quantify the correlation between these behaviors [54].

that challenges its control logic. We term this phenomenon *Decision Uncertainty* and define it operationally based on the system's high-level output:

Decision Uncertainty: *"The likelihood that the autonomous control system takes inconsistent, unsteady, or even contradicting control decisions (i.e., movement instructions) in a short period of time or specific section of the mission. Symptoms may include deviation from the optimal path, instability, abnormal and hazardous frequent changes in the movement pattern, and even crashing into nearby obstacles".*

Intuitively, we characterize this behavioral uncertainty by analyzing the stability and consistency of the UAV's planned path (its high-level control signals), rather than its internal parameters. This black-box, output-focused definition is chosen for its potential to enable real-time monitoring using readily available data, regardless of the specific control algorithm being used.

Flight Safety. While UAV safety encompasses many requirements, this study focuses specifically on the reliability of the obstacle avoidance system during autonomous flight. This includes two primary requirements: (1) the UAV must avoid crashing into obstacles, and (2) it must maintain a *safe distance* from surrounding obstacles to prevent potential collisions in the event of unforeseen environmental changes (e.g., a gust of wind). While previous work suggested a fixed threshold of 1.5m as a general safety margin [58], a more nuanced assessment is often required. Therefore, for this study, the safety of each flight was assessed manually by human experts, as detailed in the methodology below.

Behavioral Non-determinism. Many components in a robotic system—including its hardware and sensors, software platform, and control algorithms—as well as the simulation environment itself, have some degree of randomness involved. Hence, non-determinism is an expected characteristic of robot behavior [58, 59]. When the exact same test case is simulated multiple times, it is common to observe slight, and sometimes considerable, differences in the resulting trajectories, decision certainty, and even the final safety outcome. This phenomenon is particularly prevalent in challenging "corner cases", where a test scenario may pass in some runs but fail in others. These are often referred to as *flaky tests* [6, 72]. Such non-deterministic corner cases are of great interest to this empirical investigation, as they represent the boundary of the system's capabilities where the relationship between uncertainty and unsafety is most critical. The methodology described in the next section explicitly addresses this non-determinism in the data generation process to ensure statistically valid results.

6.1.2 Methodology: Data Generation and Labeling

A prerequisite for this investigation was a large and diverse dataset containing a sufficient number of flights across all four categories shown in Figure 6.1: safe, unsafe, certain, and uncertain. To generate this data, we used an extended version of the

Surrealist framework introduced in Chapter 4. While the original implementation of Surrealist effectively generates realistic and challenging test cases for obstacle avoidance, its search process is primarily focused on safety violations. To enable a robust correlation analysis, we needed to generate a dataset that also featured a rich diversity of uncertain behaviors.

To achieve this, we extended the Surrealist framework by modifying its search process. The search algorithm was expanded to not only adjust obstacle positions but also their size (length and width) and rotation, significantly increasing the diversity of the generated environmental configurations. More importantly, the fitness function guiding the search was enhanced to prioritize test cases that exhibit **non-deterministic flight trajectories**. The rationale is that scenarios producing high variability across multiple runs are more likely to be at the edge of the system's decision-making capabilities and thus more likely to exhibit uncertainty. Formally, the new distance function minimized by the search was defined as:

$$
\begin{aligned}
sum\_dist &= \min_{p \in \text{trj.points}} \sum_{o \in \text{obs}} d(p, o) \\
ave\_dtw &= \frac{1}{n} \sum_{t \in \text{trjs}} dtw(t,\ ave\_trj) \\
distance &= \begin{cases} sum\_dist \text{ - } ave\_dtw & \text{if } ave\_dtw > MAX\_DTW \\ sum\_dist & \text{otherwise} \end{cases}
\end{aligned}
\tag{6.1}
$$

Here, *sum_dist* is the original metric from Surrealist that rewards proximity to obstacles, while the new term, *ave_dtw*, measures the average DTW distance [14] between individual flight trajectories and the average trajectory over n runs. By rewarding high values of *ave_dtw*, the search is explicitly guided towards generating flaky, non-deterministic scenarios.

Using this extended framework, we generated a large pool of test cases by starting from 6 distinct initial seed scenarios, each defining different initial positions for two box-shaped obstacles (see Figure 6.2, top, for 3 examples). To ensure diversity, each of these 6 seeds was used as the starting point for an independent search process. During each search, one of the obstacles was held fixed, while the other was iteratively resized, rotated, and moved to optimize the fitness function. To account for non-determinism, each generated test case was concurrently executed 9 times, with the resulting behavioral diversity immediately visible in the flight path differences (as shown in the bottom plots of Figure 6.2). This entire campaign resulted in approximately 5,000 flight logs across the 6 datasets. These datasets were then compared, and the most diversified one—consisting of 107 unique test cases and 956 total flight logs—was selected for the in-depth analysis presented in the remaining of this chapter.

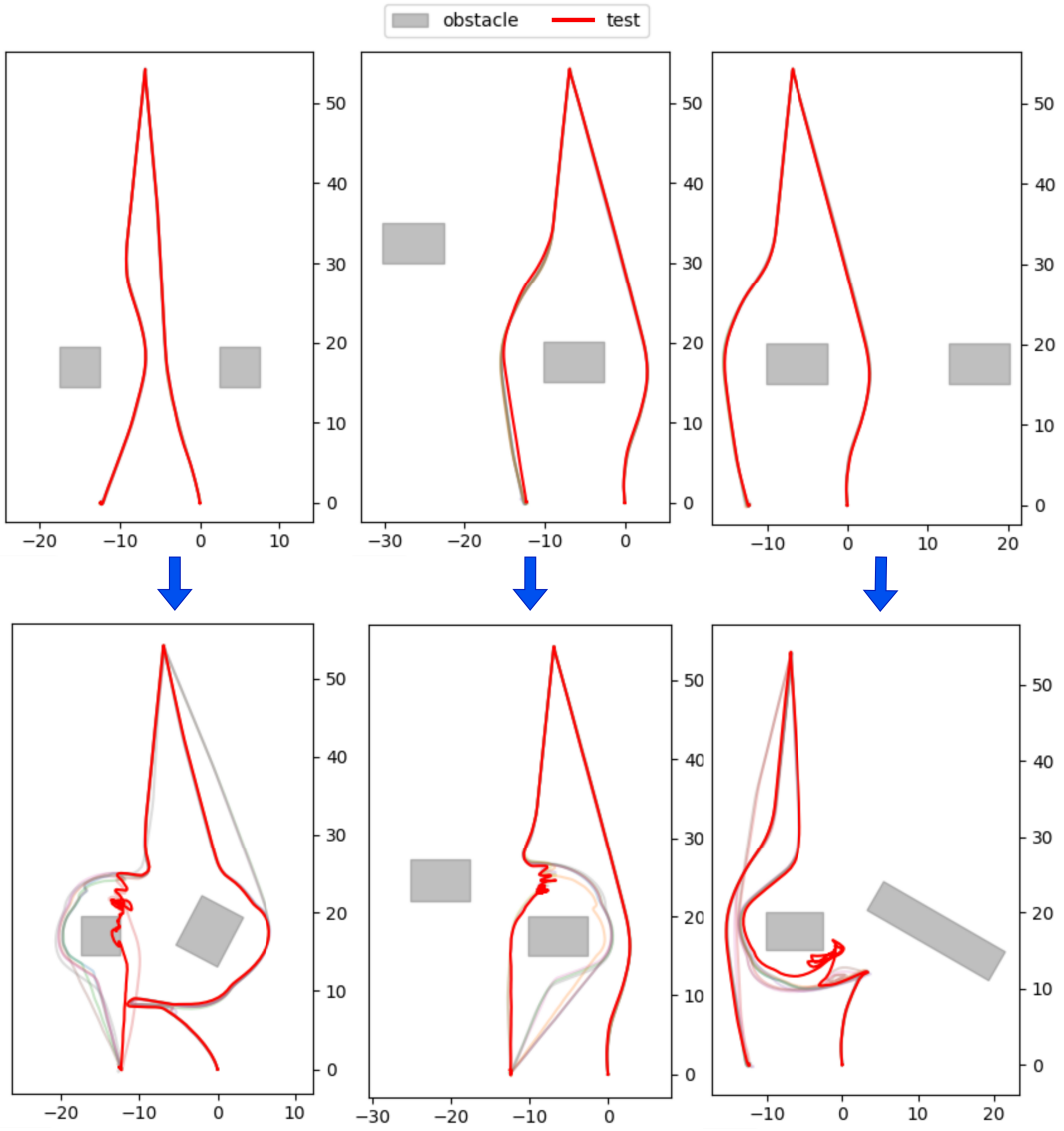

Figure 6.2. Seed test cases (top) used to initialize the search and sample generated tests (bottom) produced by the extended Surrealist. The diverse and non-deterministic trajectories in the bottom are directly related to the enhanced fitness function [54].

Table 6.1 summarizes the three datasets used in our study. The dataset generated using the new diversity-focused search (Extended Surrealist) was split into two subsets: 1) a large *train_ds* set, consisting of 86 test cases (767 flights) from the middle iterations of the search process, was reserved for training the anomaly detection models detailed in the next section. 2) *test1_ds*, composed of 21 test cases (189 flights) from the initial (first 9) and final (last 12) iterations to ensure dissimilarity from the training set, was

manually labeled and analyzed in this empirical study. A second, independent dataset from the original *Surrealist* study [58], *test2_ds*, was also included in this empirical study. This set, consisting of 53 test cases (542 flights) generated with a safety-focused fitness function, allows us to validate the generalizability of our findings across different test generation strategies.

Table 6.1. The experimental datasets. *train_ds* was used for training the anomaly detector, while *test1_ds* and *test2_ds* were manually labeled and used for the correlation analysis and monitor evaluation.

Source	**Dataset**	**#Tests×Exec.**	**#Flights**	**#<1.5m**
Extended Surrealist	*train_ds*	86×9	767	325 (42%)
	test1_ds	21×9	189	42 (22%)
Original Surrealist [58]	*test2_ds*	53×10	542	199 (37%)

To establish a ground truth for the correlation analysis, *test1_ds* and *test2_ds* were manually labeled by three validators. A custom labeling tool was implemented to display the trajectory plot for each flight. Based on this visualization, each validator assigned two independent labels according to predefined guidelines:

- **Safety Label (Safe/Unsafe):** A flight was labeled *Unsafe* if the UAV's minimum distance to an obstacle fell below 1 meter, or if it navigated a risky path between 1 and 3 meters from an obstacle that indicated a high collision risk. All other flights were labeled *Safe*.
- **Certainty Label (Certain/Uncertain):** A flight was labeled *Uncertain* if it exhibited erratic, hesitant, or inconsistent movements in a segment of its path. Flights with clear, smooth, and direct trajectories were labeled *Certain*.

To ensure the objectivity and reliability of the ground truth, the final label for each flight was determined by a majority vote among the three validators.

6.1.3 Findings: Quantifying the Correlation

The analysis of the two manually labeled datasets, *test1_ds* and *test2_ds*, revealed a moderate but significant correlation between decision uncertainty and flight safety. The confusion matrices detailing the relationship for both datasets are presented in Table 6.2. A key observation is the difference between the datasets themselves: *test2_ds*, generated using the original Surrealist implementation, contains a higher proportion

of both unsafe (25.5%) and uncertain (30.6%) flights. In contrast, *test1_ds*, generated with the extended Surrealist aimed at fostering greater diversity, has fewer unsafe (13.2%) and uncertain (16.9%) cases, but a higher proportion of borderline scenarios, as evidenced by a higher disagreement rate among the validators (10.5% vs. 6.5%) during the manual labeling process.

Table 6.2. The confusion matrices illustrating the relationship between the manually labeled Safety and Certainty states for the two test datasets.

test1_ds

Unsafe	**Uncertain**		Sum
	True	False	
True	16 (8.5%)	9 (4.8%)	25 (13.2%)
False	16 (8.5%)	148 (78.3%)	164 (86.8%)
Sum	32 (16.9%)	157 (83.1%)	189 (100%)

test2_ds

Unsafe	**Uncertain**		Sum
	True	False	
True	123 (22.7%)	15 (2.8%)	138 (25.5%)
False	43 (7.9%)	361 (66.6%)	404 (74.5%)
Sum	166 (30.6%)	376 (69.4%)	542 (100%)

As shown in Table 6.3, there is a high level of **Agreement Accuracy** between the two labels across both datasets (86.8% and 89.3%), indicating that in the vast majority of cases, flights were either both safe and certain, or both unsafe and uncertain. To delve deeper into this relationship, we analyzed two key conditional probabilities.

First, we investigated the likelihood that an uncertain decision would lead to an unsafe state, represented by $p(\text{unsafe} \mid \text{uncertain})$. The results show that this probability ranges from **50.0%** in *test1_ds* to **74.1%** in *test2_ds*. This provides strong evidence that observable decision uncertainty is a significant risk factor. However, it also reveals that a notable portion of uncertain behaviors—ranging from 26% to 50%—do not ultimately result in a safety violation. In these cases, the UAV exhibits erratic behavior but eventually recovers and finds a safe path.

Table 6.3. Statistical metrics quantifying the relationship between the Safety and Certainty labels.

Metric	***test1_ds***	***test2_ds***
Agreement Accuracy (%)	86.8%	89.3%
$p(\text{unsafe} \mid \text{uncertain})$	50.0%	74.1%
Confidence Interval	[33.6, 66.4]%	[66.9, 80.2]%
$p(\text{uncertain} \mid \text{unsafe})$	64.0%	89.1%
Confidence Interval	[44.5, 79.8]%	[82.8, 93.3]%

Finding 8 (Uncertainty as a Risk Indicator): Up to 74% of uncertain UAV decisions lead to unsafe flight states shortly thereafter. This confirms that decision uncertainty is a strong indicator of potential safety violations, although a significant fraction of uncertain events are not ultimately related to unsafety.

Second, we investigated the inverse relationship: the likelihood that an unsafe state was preceded by observable uncertainty, or $p(\text{uncertain} \mid \text{unsafe})$. Here, the correlation was even stronger, with the probability ranging from **64.0%** in *test1_ds* to a striking **89.1%** in *test2_ds*. This finding is crucial, as it suggests that the vast majority of unsafe behaviors have a detectable, observable precursor in the form of erratic control decisions. However, it also implies that a non-negligible portion of unsafe states (ranging from 11% to 36%) occur in flights where no significant uncertainty is evident. These cases represent a different class of failures, where the UAV may operate with high certainty but still follow a path that leads to a safety violation.

Finding 9 (Uncertainty as a Precursor to Failure): Up to 89% of unsafe states exhibit significant signs of decision uncertainty before the incident occurs, indicating that most safety violations have a detectable behavioral precursor. However, over 11% of unsafe states occur without any prior signs of uncertainty.

The 95% Wilson confidence intervals reported in Table 6.3 confirm the reliability of these conclusions. Despite the wider intervals for the smaller *test1_ds* dataset, the ranges support the core findings. This empirical study successfully established a quantifiable, moderate-to-strong link between observable decision uncertainty and safety violations. The correlation is significant enough to motivate and justify the development of a runtime safety monitor based on uncertainty detection, while also clarifying the inherent limitations of such an approach due to the cases where the two phenomena do not align.

6.2 Superialist: A Black-Box Runtime Safety Monitor

Motivated by the empirical findings establishing a clear link between observable decision uncertainty and safety violations, this section introduces **Superialist**. This novel, black-box runtime safety monitor is developed to leverage this correlation. Superialist is designed to detect decision uncertainty in real-time by identifying anomalies in a robot's high-level control signals, thereby providing an early warning of potentially unsafe behavior.

6.2.1 Monitoring Target: The PX4 Autonomous Navigation System

The system used for this investigation is a UAV running the PX4 software stack in *mission mode*, which enables fully autonomous flight. As illustrated in Figure 6.3, the navigation architecture consists of two primary, collaborating modules: *PX4-Autopilot*, which oversees the main flight control, and *PX4-Avoidance*, a separate module responsible for intelligent, camera-based obstacle detection and avoidance [13].

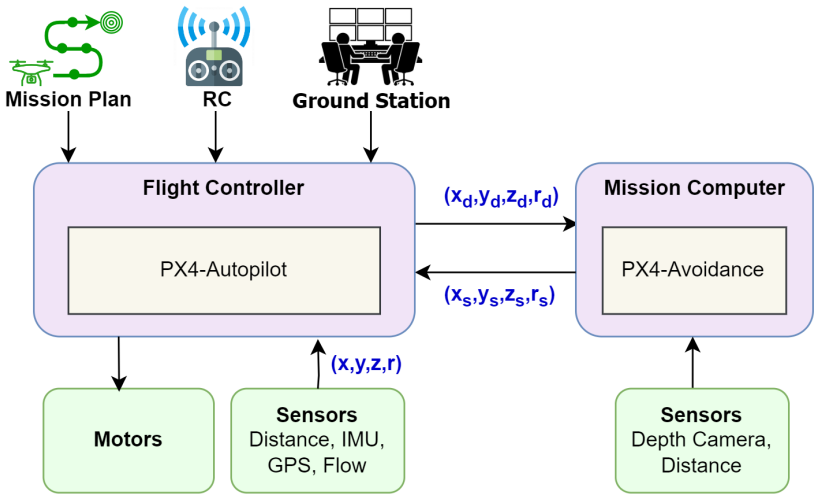

Figure 6.3. The PX4 platform architecture for autonomous navigation. The Flight Controller (running PX4-Autopilot) communicates with a Mission Computer (running PX4-Avoidance) to ensure a safe flight path [54].

During a mission, the system generates several key streams of trajectory data, which are recorded in the flight logs:

- *Desired (Blind) Waypoints* ($\langle ts, x_d, y_d, z_d, r_d \rangle$): At each timestep ts, the PX4-Autopilot calculates an optimal waypoint to reach the next mission point. This calculation is "blind" as it does not consider any immediate obstacles.

- ***Safe (Control) Waypoints*** ($\langle ts, x_s, y_s, z_s, r_s \rangle$): The PX4-Avoidance module receives the desired waypoints and, using data from its onboard cameras, checks for potential collisions. If an obstacle is detected, it updates the waypoint to a new, safe location. This stream of safe waypoints represents the final, high-level control signal that the UAV is commanded to follow.

- *Local Position* ($\langle ts, x, y, z, r \rangle$): This is the UAV's actual position in space as estimated from its sensors (e.g., GPS, IMU). This represents the resulting flight trajectory.

Superialist is designed to monitor the **Safe (Control) Waypoints** stream, as this directly reflects the final decisions of the obstacle avoidance system before they are executed by the low-level controllers.

Our preliminary experiments revealed that the most reliable and information-rich signal for detecting decision uncertainty was the **commanded heading angle (r_s)**, or yaw, from the safe waypoints. This is because the UAV used in this study, like many commercial drones, relies on a single, front-facing stereo camera for obstacle detection. To perceive a complex environment, the UAV must physically rotate its body by changing its heading. During periods of uncertainty, the PX4-Avoidance module often generates rapid, inconsistent, and oscillating heading commands as it searches for a safe path, making this single variable a strong indicator of anomalous behavior.

As shown in Figure 6.4, this decision uncertainty is far more evident in the control signals (left) than in the resulting flight trajectory (right). The erratic, back-and-forth pattern in the commanded waypoints around point C is a clear signal of instability, whereas the actual flight path is much smoother due to the physical inertia of the UAV. By focusing on these raw control signals, Superialist can detect signs of trouble earlier and more clearly.

6.2.2 Core Approach: Anomaly Detection on Control Signals

The core approach of Superialist is to treat the problem of detecting decision uncertainty as a task of **unsupervised anomaly detection**. The fundamental idea is to build a model of a system's *normal* or *nominal* behavior and then identify any significant deviations from this model at runtime.

A key design principle of Superialist is its **black-box nature**. It does not require any internal knowledge of the robot's control algorithms or access to its low-level state. Instead, it operates exclusively on the high-level control signals that the navigation system produces—specifically, the sequence of safe waypoints it commands the robot to follow. This approach has several significant advantages:

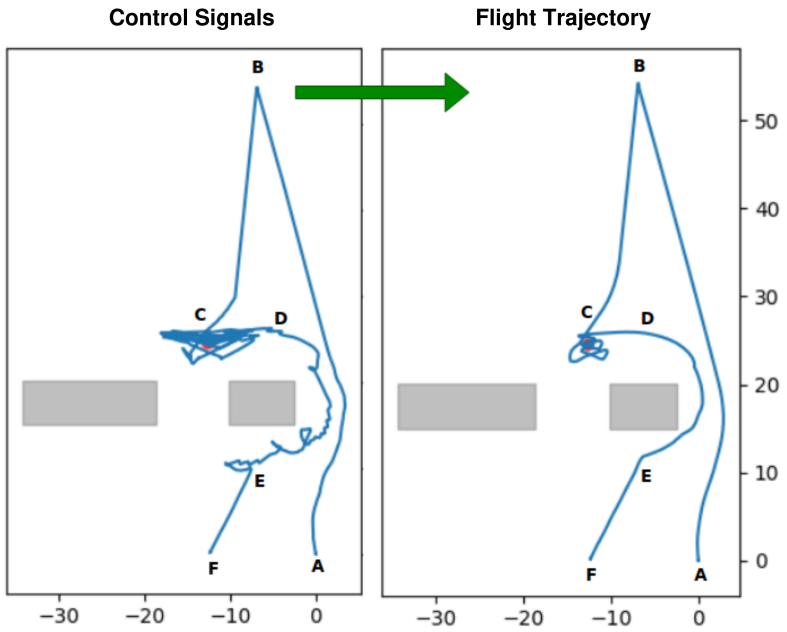

Figure 6.4. A comparison of the high-level control signals (left) and the resulting flight trajectory (right). The erratic pattern of the control signals around point C is a clear indicator of decision uncertainty, which is less apparent in the smoother, final flight path [54].

- **Lightweight:** By focusing on a minimal set of high-level data, the monitor is computationally inexpensive, making it suitable for deployment on resource-constrained onboard computers.
- **Generalizable:** Because it is not tied to any specific implementation, the same monitoring approach can be applied to different navigation algorithms or even different robotic platforms with minimal modification.
- **Non-invasive:** The monitor can be added as a supervisory layer to an existing system without requiring any changes to the core control software.

6.2.3 Implementation with Autoencoders

Superialist is implemented using a **convolutional autoencoder**, a type of neural network well-suited for unsupervised anomaly detection in time-series data [106]. An autoencoder is trained to reconstruct its own input. It consists of two parts: an *encoder*, which compresses the input data into a lower-dimensional representation (the latent space), and a *decoder*, which attempts to reconstruct the original data from this

compressed representation. To demonstrate the feasibility of using autoencoders for uncertainty detection in our context, we employ a convolutional autoencoder with a simple, common architecture for anomaly detection in time-series data [53], depicted in Figure 6.5. The following subsections detail the data processing pipeline and the model's training and inference process.

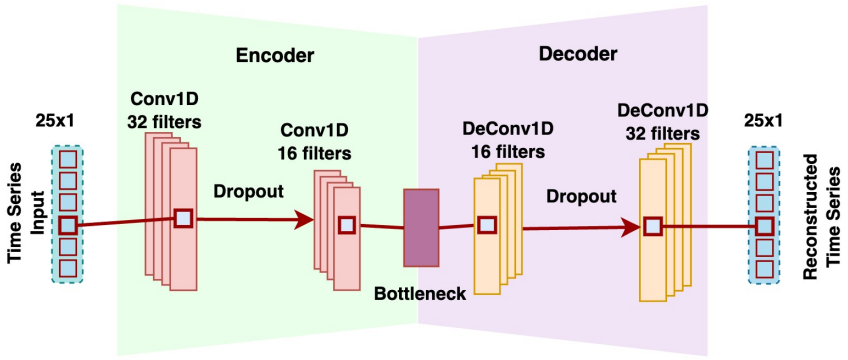

Figure 6.5. The convolutional autoencoder architecture used in Superialist. The model takes a time-series window of heading commands as input and is trained to reconstruct it. High reconstruction loss indicates an anomalous or uncertain behavior pattern [54].

Data Preprocessing

To prepare the data for the autoencoder, the continuous stream of commanded heading angles is segmented into fixed-length, overlapping time windows (a window length of 5 seconds with a 2.5-second overlap was used in our experiments). Each window represents a snapshot of the UAV's recent decision-making process.

Training and Inference

The autoencoder is trained in an unsupervised manner. The training dataset consists exclusively of data from **nominal flights**—that is, flights that are known to be both safe and certain. This is achieved by filtering the *train_ds* to include only time windows where the UAV is operating far from any obstacles. By training only on normal data, the autoencoder learns to accurately reconstruct the patterns associated with stable and predictable navigation.

During runtime **inference**, the trained autoencoder is fed a continuous stream of time windows from the live flight. When the UAV is operating normally, the input data matches the patterns the autoencoder has learned, and it can reconstruct the time window with a very low **reconstruction loss** (e.g., mean squared error). However,

when the UAV enters a state of decision uncertainty, the pattern of heading commands deviates significantly from the normal patterns seen during training. The autoencoder struggles to reconstruct this unfamiliar data, resulting in a high reconstruction loss.

Superialist detects an anomaly—and thus, a state of decision uncertainty—by setting a threshold on this reconstruction loss. If the loss exceeds the predefined threshold for a sustained period, a warning is triggered, indicating that the system has entered an unstable and potentially unsafe state.

6.3 Evaluation of Superialist's Predictive Capabilities

To evaluate the effectiveness of the Superialist monitor, the trained autoencoder was rigorously assessed using the two separate, manually labeled test datasets, *test1_ds* and *test2_ds*. The evaluation was designed to answer two key questions:

- **RQ_2 (Uncertainty Detection):** *How accurately can Superialist detect the ground-truth decision uncertainty labels?*
- **RQ_3 (Unsafety Prediction):** *How well does this uncertainty detection serve as a proxy for predicting upcoming unsafe states?*

6.3.1 Defining the Anomaly Threshold

Establishing an optimal threshold for the reconstruction loss is a crucial element of our methodology, as this threshold serves to distinguish between normal and anomalous UAV behavior (i.e., certain and uncertain decisions). We empirically observed that during flight, a sudden peak in reconstruction loss is expected and normal when initiating common maneuvers, such as intentional turns at mission waypoints or switching flight modes. To accommodate this, we enhanced our methodology by computing the *mean reconstruction loss* over $n = 4$ consecutive time windows instead of considering single windows. This modification provides a more precise representation of the UAV's operational condition, mitigating minor, temporary fluctuations in reconstruction loss and thus reducing false positives.

Since there is no established standard for an "acceptable" level of decision uncertainty in UAVs, the threshold had to be determined empirically from our datasets. To do this, we relied on the histogram of the reconstruction loss derived from the nominal training data. Figure 6.6 represents the frequency distribution of different loss levels observed during nominal UAV operations. As depicted in the figure (note the logarithmic scale), the majority of time windows produced a reconstruction loss below 0.05. However, there is a significant drop in frequency for values exceeding 0.2. Given the

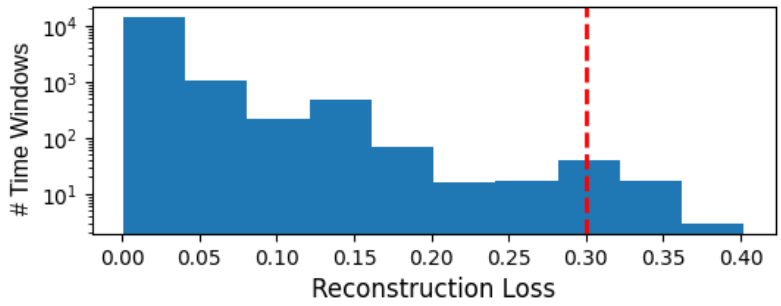

Figure 6.6. A histogram of the reconstruction loss for the nominal (safe and certain) training data. The threshold for anomaly detection was set at 0.3, capturing significant deviations from this normal distribution [54].

expectation that a small number of anomalies might have survived the filtering of the training data, we set the anomaly detection (i.e., uncertainty detection) threshold at $\theta = 0.3$. This threshold was also chosen to favor a higher True Positive Rate (correctly detected uncertainties), accepting a potential increase in the False Positive Rate (false alarms) as a trade-off, given the critical importance of early failure detection.

This data-driven approach ensures the threshold is grounded in the system's nominal behavior, allowing us to effectively identify and capture anomalous behaviors that deviate significantly from the norm. Using this threshold, only 39 time windows in the nominal training set (belonging to 4 different flight logs) were flagged as uncertain. Manual analysis confirmed that these were, in fact, instances of true uncertainty that had not been filtered from the training data, resulting in a 0% false positive rate on the training set.

6.3.2 Performance in Detecting Decision Uncertainty (RQ_2)

The primary task of the monitor is to identify anomalous patterns in the UAV's control signals that correspond to decision uncertainty. Table 6.4 presents the confusion matrices for this task. The evaluation confirmed that Superialist is highly effective. As shown in Table 6.5, the model achieved a high precision of **90.0%** and **95.7%** on the two test sets, confirming that when Superialist raises an alarm for uncertainty, it is highly likely to be correct.

The recall was also strong, at **84.4%** and **93.4%**, indicating that the monitor successfully identified the vast majority of all actual instances of uncertain behavior present in the datasets. The slightly lower recall for *test1_ds* can be attributed to the greater diversity and subtlety of the uncertain behaviors in that dataset. Overall, with F1-scores of 87.1% and 94.5%, the results validate that the autoencoder-based approach pro-

vides a reliable, black-box solution for monitoring and detecting decision uncertainty in real-time.

Table 6.4. Uncertainty Detection Confusion Matrices

test1_ds

Predicted	**Uncertain**	
	True	False
True	27	3
False	5	154

test2_ds

Predicted	**Uncertain**	
	True	False
True	155	7
False	11	369

Table 6.5. Performance metrics for the Superialist monitor on the two test datasets, evaluated against both the ground-truth Uncertainty and Safety labels.

Target Label	**Dataset**	**Precision**	**Recall**	**F1-Score**
Uncertainty	*test1_ds*	90.0%	84.4%	87.1%
	test2_ds	95.7%	93.4%	94.5%
Unsafety	*test1_ds*	43.3%	52.0%	47.3%
	test2_ds	74.1%	87.0%	80.0%

Finding 10 (Uncertainty Detection): Superialist effectively detects anomalies in the high-level control signals of UAVs, enabling cost-effective real-time identification of decision uncertainties. With a precision of up to 95.7% and a recall of up to 93.4%, the autoencoder approach stands as a reliable black-box solution for monitoring UAV decision-making.

6.3.3 Performance in Predicting Unsafe States (RQ_3)

The second, more challenging goal was to evaluate how well this detected uncertainty could predict actual safety violations. The confusion matrices for this task are shown in Table 6.6. As detailed in Table 6.5, when the monitor's predictions were evaluated against the ground-truth safety labels, there was a notable decline in performance, particularly in precision. The precision for predicting unsafety was **43.3%** for *test1_ds* and **74.1%** for *test2_ds*.

This result is not a failure of the monitor itself, but rather a direct reflection of the underlying correlation established in the empirical study. As established in Finding 1, a significant portion of uncertain events do not lead to unsafe states. This phenomenon is

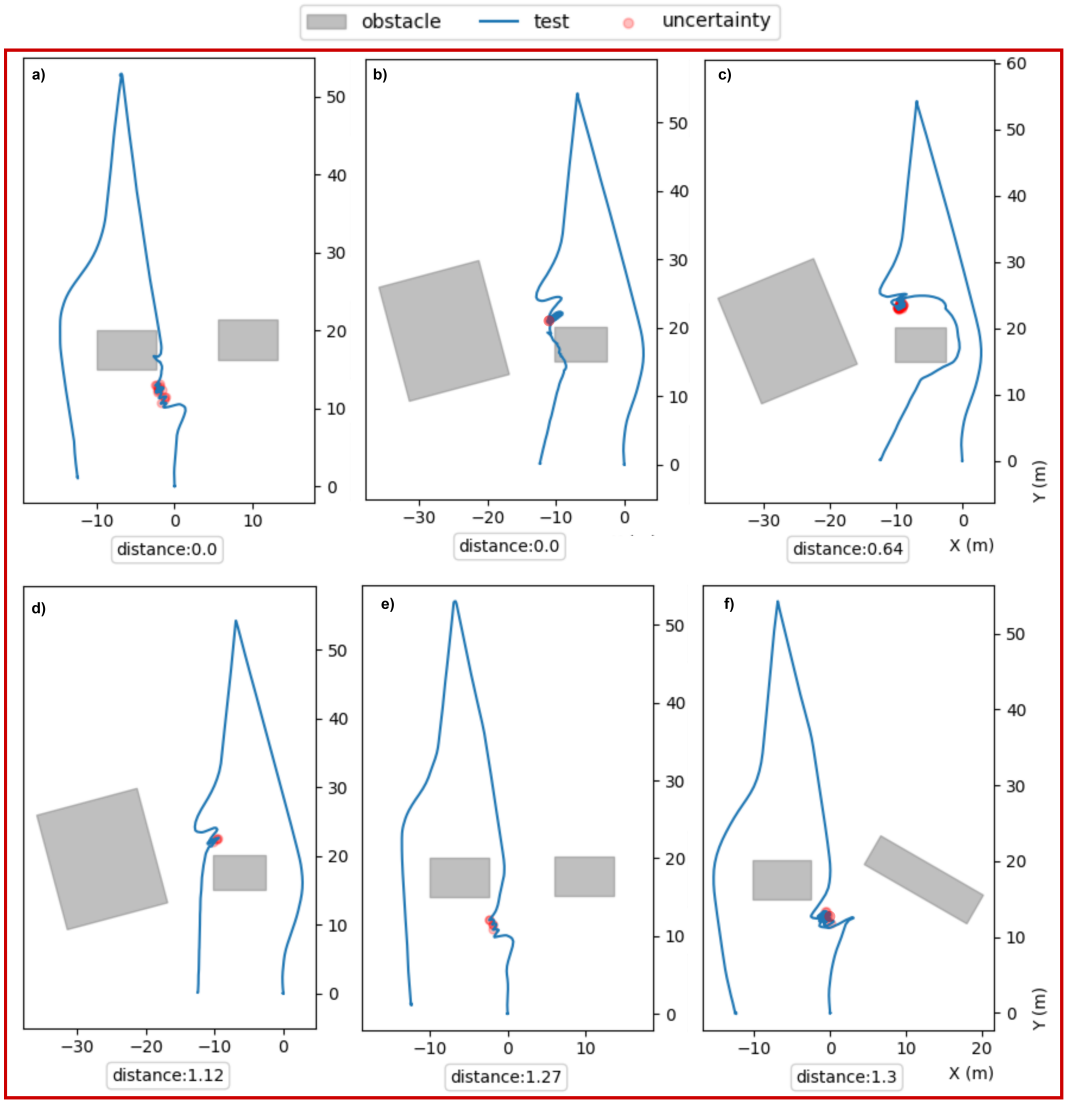

Figure 6.7. Qualitative evaluation of Superialist (Part 1 of 2). Flight trajectories overlaid with Superialist's uncertainty detections (red dots). This set illustrates the **True Positive** cases (a–f), where the monitor successfully identified decision uncertainty preceding the actual unsafe behavior [54].

visually demonstrated in Figure 6.8 (plots i-l), which showcase flights labeled as uncertain but ultimately safe. In these instances, Superialist correctly identifies uncertainty in the UAV's decisions, yet the drone navigates safely in subsequent seconds. These correct uncertainty detections register as false positives when evaluated against the safety label, thus lowering the precision. Indeed, the maximum achievable precision

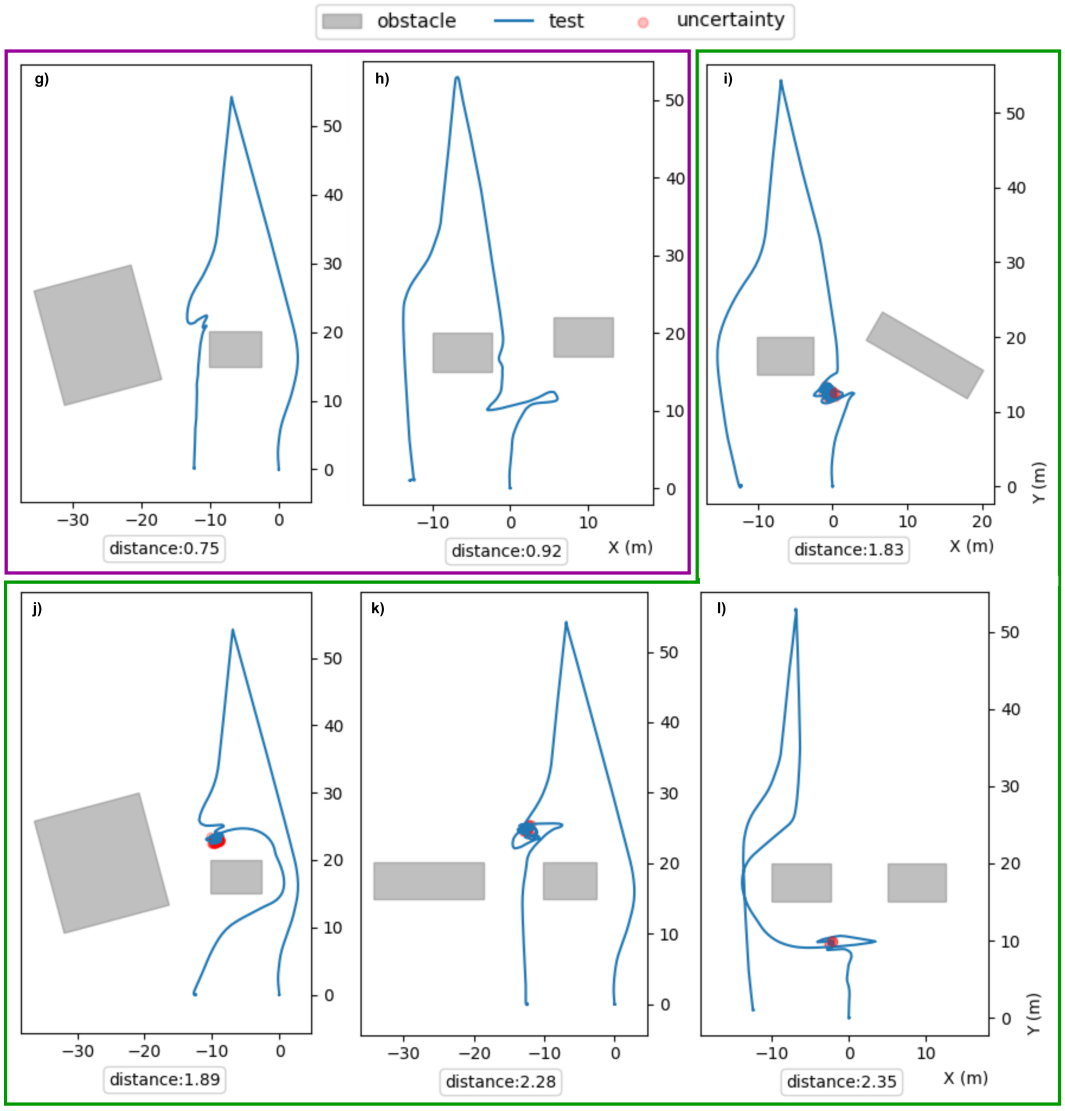

Figure 6.8. Qualitative evaluation of Superialist (Part 2 of 2). Continued flight trajectories overlaid with uncertainty detections (red dots). This set illustrates the misclassified cases: **False Negatives** (g, h), where unsafe behavior occurred without any prior uncertainty warnings, and **False Positives** (i–l), where the system exhibited uncertainty but ultimately navigated safely [54].

was bounded by the empirical correlation of $p(\text{unsafe} \mid \text{uncertain})$, which was 50.0% and 74.1% for the two datasets.

The recall, however, remained relatively high at **52.0%** and **87.0%**, which aligns with Finding 2—that the majority of unsafe states are indeed preceded by detectable

uncertainty. This is illustrated in Figure 6.7 (plots a-f), which depict flights manually labeled as both unsafe and uncertain, where Superialist successfully identifies the uncertainty prior to the safety violation. However, the recall is not perfect because, as noted in Finding 2, some unsafe states occur without any prior signs of uncertainty. Figure 6.8 (plots g, h) provides examples of such cases, where the UAV follows a confident but incorrect path into an unsafe state.

Table 6.6. Unsafety Prediction Confusion Matrices

test1_ds

Predicted	**Unsafe**	
	True	False
True	13	17
False	12	147

test2_ds

Predicted	**Unsafe**	
	True	False
True	120	42
False	18	362

Finding 11 (Unsafety Prediction): Anomalies in control signals are moderately effective for predicting unsafe drone states, with up to 74.1% precision and 87.0% recall. The performance is inherently limited by the moderate correlation between uncertainty and unsafety, confirming that even a perfect uncertainty detector would perform suboptimally for direct unsafety prediction.

6.3.4 The Early Warning Advantage and Practical Implications

Despite the limitations in precision for unsafety prediction, a critical contribution of this monitoring approach is its ability to provide a significant **early warning**. An analysis of the true positive predictions revealed that, on average, the first uncertainty warning was raised **50.2 seconds** before the UAV reached a critical zone (less than 1m from an obstacle) in *test1_ds*, and **42.4 seconds** in *test2_ds*. At the moment of the first alert, the UAV was, on average, still a safe 3.45m and 3.72m away from the nearest obstacle.

This finding is of paramount practical importance. An early warning of tens of seconds provides a vast time window for a human supervisor to be alerted and intervene, or for an automated self-healing or fail-safe mechanism to be triggered, potentially averting a crash or other safety violation long before it becomes critical. Figure 6.9 illustrates this, showing how the monitor flags long periods of uncertainty well in advance of the UAV reaching the unsafe area around the obstacles.

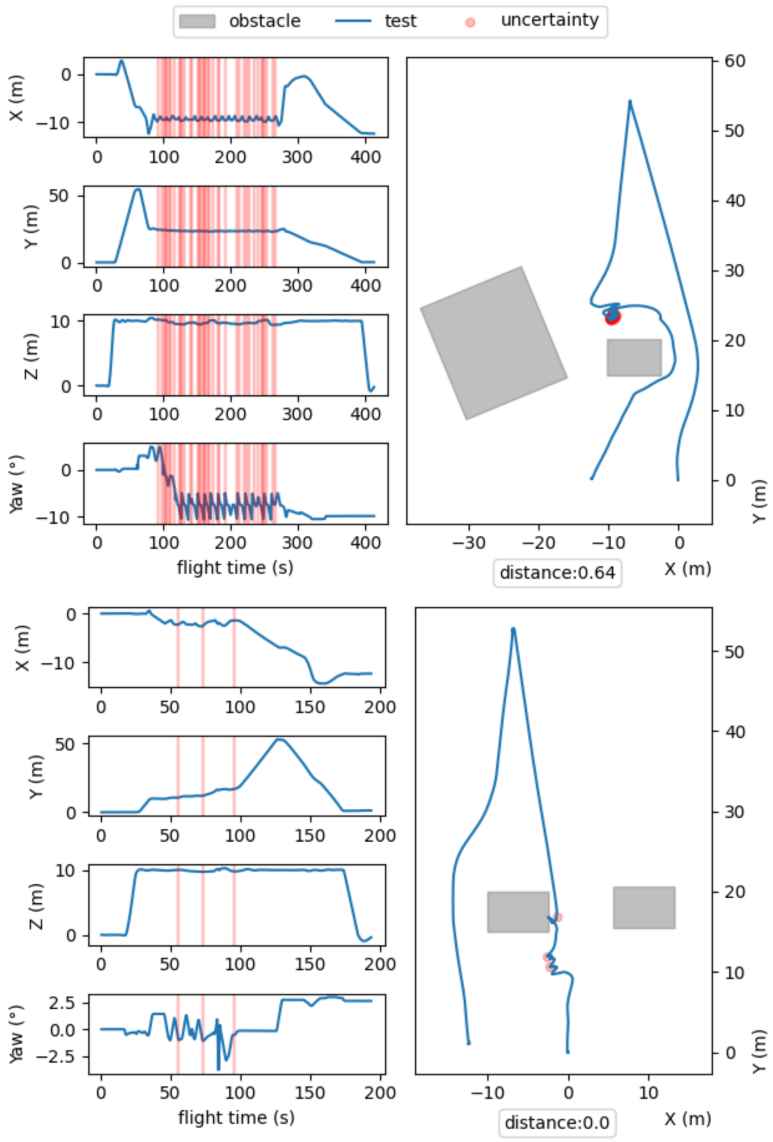

Figure 6.9. Examples of uncertainty detected by Superialist over the course of two flights. The red highlights in the time-series plots and the red dots on the trajectory plots show where uncertainty was detected. In both cases, the warnings are triggered long before the UAV reaches the critical, unsafe area near the obstacles [54].

Finding 12 (Early Warning): Detecting decision uncertainty provides a feasible and effective early warning for safety violations. On average, the monitor detects uncertainty 42-50 seconds before a critical incident, offering a substantial time window for human or automated intervention.

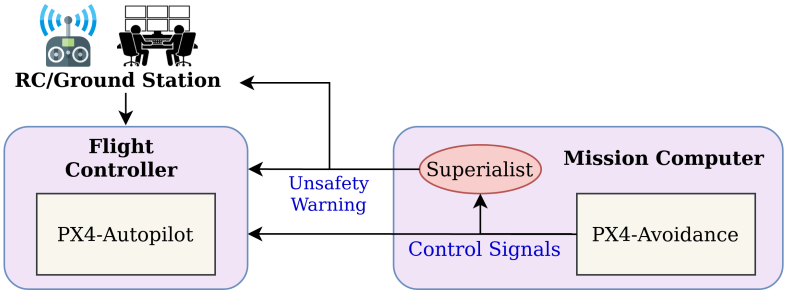

Figure 6.10. A proposed deployment architecture for Superialist. The monitor runs as a supervisory process, analyzing control signals in real-time and issuing safety warnings to the main flight controller and ground station if sustained uncertainty is detected [54].

This wide reaction window has direct practical applications for enhancing overall system safety. It is particularly relevant in the context of emerging regulations for commercial drone operations, which often require remote human pilots to monitor multiple simultaneous flights [78]. An automated alert from an uncertainty-based monitor could effectively draw a pilot's attention to a high-risk situation long before it becomes critical.

Beyond manual supervision, the signal from the monitor could be integrated into a Dynamic Safety Assurance (DSA) framework [93], serving as a trigger for automated self-healing mechanisms or configuration changes to maintain system safety. A proposed deployment architecture for integrating such a monitor into the PX4 ecosystem is shown in Figure 6.10. In this architecture, the monitor runs as a non-invasive supervisory process on the mission computer, analyzing control signals from the obstacle avoidance module and issuing safety warnings to the main flight controller (to trigger potential fail-safe behaviors) and ground station (to inform the operator for potential intervention).

6.4 Chapter Summary and Discussion

This chapter presented a two-part investigation into the runtime safety monitoring of autonomous robotic systems. The work began with a large-scale empirical study that successfully established a quantifiable, moderate-to-strong correlation between observable *Decision Uncertainty* and safety violations in UAV navigation. The findings confirmed that while most unsafe states are preceded by detectable uncertainty, not all uncertain behaviors lead to failure.

Building on this insight, the chapter introduced **Superialist**, a novel, black-box runtime monitor based on an autoencoder. The evaluation demonstrated that Superialist is highly effective at its primary task of detecting decision uncertainty, achieving high

precision and recall. While its performance as a direct unsafety predictor is inherently limited by the underlying correlation, the monitor's ability to provide warnings 40-50 seconds in advance highlights its practical value as an early warning system.

6.4.1 Future Directions and Broader Applicability

The principles and positive results of this study open several promising avenues for future research, particularly in generalizing the approach and integrating it into a holistic safety framework.

Generalizability to Other Robotic Platforms A key strength of the black-box monitoring approach is its potential for generalizability. The core concept of detecting anomalies in high-level control signals is not specific to UAVs. Most mobile robots share a similar hierarchical mobility architecture, where a high-level navigation module computes and sends target commands (e.g., desired velocity or waypoints) to a lower-level locomotion controller. This is true for the ANYmal quadruped, where the navigation stack sends high-level commands to the locomotion controller, just as PX4-Avoidance sends commands to PX4-Autopilot in the UAV use case.

Significantly, during the industrial study at ANYbotics discussed in Chapter 5, a similar behavioral uncertainty was observed: in some corner cases, the ANYmal robot would hesitate or exhibit erratic movements when navigating through narrow gaps. This suggests that the same underlying principles of uncertainty detection could be applied. Future work should explore adapting this monitoring approach to the ANYmal platform, likely by training an autoencoder on its nominal ROS-based control commands to detect analogous patterns of indecision.

Closing the Loop: From Runtime Monitoring to Test Generation A powerful future direction lies in creating a feedback loop that connects runtime safety assurance with pre-deployment testing. The data gathered by a monitor like Superialist is invaluable. When an uncertain or unsafe event is detected and logged during a real-world mission, this log file (containing sensor data and trajectories) can be fed directly back into the *Surrealist* framework as a new seed scenario.

This *real-to-sim* capability, which was a key request from industrial partners, would enable a continuous improvement cycle for the entire development process. Engineers could:

- **Replicate Failures:** Use Surrealist's replication phase to automatically reproduce the exact real-world failure in simulation for efficient debugging.

- **Generate Regression Tests:** Use the challenging scenario generation phase to create a robust test suite around the failure, ensuring the bug fix is effective in similar situations.
- **Benchmark Algorithms:** Use this new, real-world-derived test suite as an objective benchmark to compare alternative algorithms or future software versions.
- **Strengthen CI/CD:** Integrate these regression tests into the CI/CD pipeline to prevent the re-introduction of the same error in future releases.

Improving Prediction with Gray-Box Monitoring Finally, while this study focused on a black-box approach for its generalizability, future research should explore gray-box or white-box monitoring to further improve the precision of unsafety prediction. A promising direction is to combine the black-box behavioral insights from control signals with internal data from the navigation algorithm (e.g., planner costmaps or state machine transitions) or real-time data from perception sensors. This fusion of data could create a more holistic and context-aware safety monitor, better-equipped to distinguish between harmless uncertainty and genuine, high-risk situations.

6.4.2 Limitations of the Study

It is important to acknowledge the limitations of this study. Regarding **internal validity**, the use of a specific autoencoder architecture may introduce model-specific biases, and the manual labeling of flight data is subject to human judgment, though this was mitigated by using multiple validators and clear guidelines. Threats to **external validity** also exist, as the findings are based on simulated UAV flights. While simulation is a best practice, the extent to which these results generalize to real-world operations with complex environmental conditions (e.g., wind) is a topic for future investigation; the study was also limited to a specific type of navigation scenario with static obstacles. Finally, concerning **conclusion validity**, the conclusions are supported by robust statistical methods, including the use of Wilson's confidence intervals and repeated experiments to account for non-determinism. Despite these limitations, this research provides a strong foundation for the development of data-driven safety measures for autonomous systems.

Chapter 7

Conclusion and Future Work

The rapid expansion of autonomous robotic systems into safety-critical domains, from industrial inspection to aerial delivery, creates an urgent need for verification and validation techniques that can keep pace with their increasing complexity. The core challenge, as established in this dissertation, is the persistent *reality gap* between high-fidelity simulation and the unpredictable nature of the physical world. This gap undermines the reliability of traditional testing methods and creates a significant barrier to assuring the safety and robustness of these intelligent systems.

This dissertation confronts this challenge by proposing and validating a comprehensive, multi-layered framework for simulation-based testing and runtime safety monitoring of autonomous robotic systems. The central thesis of this work is that the reality gap can be bridged through a principled, data-driven approach that grounds simulation-based testing in real-world operational data and complements pre-deployment verification with a layer of runtime assurance. Through studies on two distinct and complex robotic platforms—Unmanned Aerial Vehicles and industrial quadrupedal robots—this work demonstrated the effectiveness of this approach, culminating in its successful adoption within a leading industrial robotics workflow.

7.1 Summary of Contributions

This dissertation makes four primary contributions to the field of software engineering for autonomous robotic systems:

1. A General-Purpose Framework for Test Automation (Aerialist) The first contribution is the design and implementation of *Aerialist*, a modular, scalable, and extensible framework for automating the lifecycle of simulation-based robotics testing. By providing a standardized test description model and abstracting away the low-level

complexities of interacting with different simulators and robotic software stacks (e.g., PX4 and ROS), Aerialist serves as a foundational platform that enables rigorous and repeatable experimentation. Its support for parallel execution on Kubernetes clusters was instrumental in making the large-scale, search-based studies in this dissertation feasible. This platform's utility and stability were further validated as it serves as the core test automation and benchmarking engine for the international UAV Testing Competition, fostering broader community research.

2. A Methodology for Realistic Test Generation (Surrealist) Building upon the Aerialist framework, the second contribution is *Surrealist*, a novel, two-phase, search-based methodology for automatically generating test cases that are both realistic and challenging. To bridge the reality gap, Surrealist first uses an adaptive greedy search to *replicate* real-world behaviors in simulation by optimizing environmental parameters to match operational log data. It then uses the same search-based approach to generate *challenging* new scenarios in the neighborhood of this replicated reality, systematically discovering safety-critical edge cases. The validation on UAVs demonstrated that this approach could both faithfully replicate real flight trajectories with high precision and automatically discover latent bugs in the PX4 navigation stack, leading to unsafe behaviors and crashes.

3. Cross-Domain Validation and Industrial Adoption The third contribution is the rigorous validation of the framework's generalizability and practical utility through a comprehensive industrial case study with ANYbotics. The successful adaptation of the framework from aerial to legged robots demonstrated its cross-domain applicability. The subsequent deployment within the ANYbotics development workflow provided strong evidence of its industrial impact, where it was used to find critical failures in proprietary navigation algorithms and provide objective benchmarks for assessing performance improvements. This initial success led to a deeper integration: the framework was extended into the company's MLOps pipeline, adapting the test generation methodology to the massively parallel Isaac Gym simulator. This extension dramatically accelerated the feedback cycle for ML engineers, reducing test suite execution from hours to minutes. This successful, multi-stage technology transfer serves as a key validation of the principles and designs proposed in this thesis.

4. A Black-Box Approach for Runtime Safety Monitoring (Superialist) Finally, recognizing that pre-deployment testing alone is insufficient, the fourth contribution is a novel approach to runtime safety assurance. This work began with a large-scale empirical study that established a quantifiable, moderate-to-strong correlation between ob-

servable *Decision Uncertainty* and impending safety violations. Based on this finding, we developed *Superialist*, a lightweight, black-box runtime monitor that uses an autoencoder to detect decision uncertainty by identifying anomalies in a robot's high-level control signals. The evaluation showed that Superialist can detect uncertainty with high precision and recall and, most importantly, provide a significant early warning—up to 50 seconds in advance of a critical event—thereby creating a crucial window for corrective action.

7.2 Future Work

The research presented in this dissertation has successfully validated a complete methodology for simulation-based testing and runtime monitoring, culminating in its successful industrial adoption. This process, however, has also illuminated several fundamental challenges and high-impact opportunities for future research that extend beyond the specific implementations detailed in the preceding chapters.

Beyond Greedy Search: The Future of Test Generation The *Surrealist* framework's use of an adaptive greedy search guided by a hand-crafted fitness function proved highly effective, generalizable, and was key to its successful adoption. Moreover, the extension of *Surrealist* to generate non-deterministic test cases (as detailed in Chapter 6) confirmed its flexibility in adapting to new test goals, provided the right fitness function can be engineered. However, this success also revealed a practical bottleneck: defining a new fitness function is a non-trivial task. As feedback from ANYbotics engineers confirmed, quantifying complex test goals—beyond simple obstacle proximity—requires deep domain knowledge and significant experimentation. For instance, prioritizing scenarios that induce path-planning failures (where the robot is unable to find a safe path), mission timeouts (where navigation is too slow), or subtle non-deterministic behaviors is difficult to encode in a simple proximity-based metric.

Now that the core concept of search-based, reality-grounded testing is established, future work is essential to explore alternative ways of guiding the test generation process. Machine learning-based approaches, including reinforcement learning (RL), are promising for learning complex test-generation policies that can explore the vast configuration space more effectively. Such methods could also be trained to optimize for multiple objectives in parallel (e.g., combining obstacle proximity, decision uncertainty, and navigation time). Furthermore, while *Surrealist* focused on finding critical failures, it did not explicitly optimize for test suite diversity. Future frameworks should incorporate diversity-preservation mechanisms to ensure the generated scenarios cover a wider range of distinct behavioral edge cases.

Test Generation for MLOps vs. DevOps While the framework was originally conceived for test automation within a traditional CI/CD (DevOps) pipeline—where test failures serve as regression gates to prevent merging faulty code—the industrial study at ANYbotics revealed a second, equally important use case. The generated test suites are not just regression checks; they are valuable assets that serve as rigorous, automated benchmarks for evaluating and comparing new algorithms. This benchmarking capability was the primary driver for the MLOps extension, which allows ML engineers to rapidly evaluate new models in Isaac Gym.

This distinction highlights a crucial future research direction: defining the different roles of test suites. While a test generator may find thousands of challenging corner cases, not all of these represent violations of hard operational requirements that should block a production release. Instead, these comprehensive suites are ideal for the MLOps pipeline, which is triggered when new ML models are trained. Here, the goal is not a simple pass/fail but to obtain a rich, objective benchmark of a model's performance across all types of scenarios—from simple to complex edge cases. A curated subset of these tests, representing critical, known failure modes, could then be promoted to the faster DevOps pipeline. This smaller suite would run on every code change (e.g., in a perception module) to quickly catch integration-related regressions, creating a more efficient, two-tiered assurance process.

Fostering a Research Community and Extending to New Domains The *Aerialist* platform and the UAV Testing Competition, established as (hopefully lasting) contributions of this thesis, are organized around the long-term, community-wide goal of fostering future research in this domain. They provide a simple, common platform to enable the comparison and benchmarking of state-of-the-art practices, helping the community identify the most promising directions to follow. The documented extension process for the ANYmal robot was intended to further help the community in adapting the platform to even more use cases. There are many interesting domains—including other ground mobile robots (e.g., TurtleBot), small indoor drones (e.g., Crazyflie), and collaborative robotic arms (e.g., UR)—that could be explored. Extending our platform to support test automation for them would highly benefit researchers by preventing them from "re-inventing the wheel".

Testing for a Complex and Dynamic World This thesis, like most foundational research in this area, intentionally simplified the testing scenarios to create a controlled experimental setup. Our environments focused on static obstacles with simple shapes and did not consider dynamic environmental factors like wind, lighting changes, or complex 3D terrain. While this approach proved effective for finding significant al-

gorithmic flaws, the next generation of autonomous systems (humanoids, self-driving cars, autonomous delivery robots) will be deployed in dynamic, human-centric environments. This creates an urgent need for verification methodologies that can handle this complexity. A huge gap currently exists between the advanced capabilities of modern robots and our ability to rigorously test their behavior. Future research must focus on generating test scenarios that include dynamic obstacles (such as other robots or humans), non-uniform and complex obstacle shapes, and variable environmental conditions, as well as developing runtime monitors for such scenarios.

Extending the Assurance Methodology to the Full Robotics Stack This dissertation focused on the navigation system as the highest level of abstraction in the robot's mobility stack, making it an ideal candidate for a black-box, system-level assurance approach. However, the industrial collaboration revealed that teams responsible for lower-level systems, such as locomotion and perception, are in equal need of automated assurance frameworks. Future work should focus on adapting the core principles of this thesis to these new domains. The hierarchical structure of robotics, where navigation issues high-level commands to locomotion, provides a clear path for generalization. The two-phase methodology (pre-deployment testing and runtime monitoring) and the *Surrealist* concept of bridging the reality gap are still highly relevant. However, applying them to domains like locomotion will require new, more refined methodologies, such as defining fitness functions based on stability metrics and joint torque limits rather than just environmental proximity.

Enhancing "Real-to-Sim" Replication and Closing the Loop The test replication phase of *Surrealist* served as a proof of concept, demonstrating the relevance and importance of grounding test generation in real-world data. The strong feedback from our industrial collaborators confirmed the high value of this concept, highlighting a clear demand for a more advanced "real-to-sim" capability. While the full development of such a feature was not in the scope of the ANYbotics case study, it represents a critical direction for future work. This would involve creating tools that can automatically generate seed scenarios from real-world sensor data, such as point clouds or recorded video, to fully automate the process of replicating and diagnosing field failures. Furthermore, future research could explore replicating more complex phenomena beyond static environments, such as variable environmental conditions (e.g., wind or lighting) or specific low-level behaviors (e.g., motor saturation or controller instability) rather than just the robot's trajectory. This would effectively *close the loop* in the assurance process, as described in Chapter 6: a runtime monitor detects a real-world anomaly, which is then automatically fed into the replication framework to generate a new, challenging test

suite for regression testing and algorithm benchmarking.

Bibliography

[1] Abdessalem, R. B., Nejati, S., Briand, L. C. and Stifter, T. [2018]. Testing vision-based control systems using learnable evolutionary algorithms, *International Conference on Software Engineering*, ACM, pp. 1016–1026.
URL: *http://doi.acm.org/10.1145/3180155.3180160*

[2] Afzal, A., Goues, C. L., Hilton, M. and Timperley, C. S. [2020]. A study on challenges of testing robotic systems, *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, pp. 96–107.

[3] Afzal, A., Katz, D. S., Le Goues, C. and Timperley, C. S. [2021]. Simulation for robotics test automation: Developer perspectives, *Conference on Software Testing, Verification and Validation*, IEEE, pp. 263–274.

[4] Afzal, A., Le Goues, C., Hilton, M. and Timperley, C. S. [2020]. A study on challenges of testing robotic systems, *International Conference on Software Testing, Validation and Verification*, IEEE, pp. 96–107.

[5] Ahn, H. [2020]. Deep learning based anomaly detection for a vehicle in swarm drone system, *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, pp. 557–561.

[6] Amini, M. H., Naseri, S. and Nejati, S. [2024]. Evaluating the impact of flaky simulators on testing autonomous driving systems, *Empir. Softw. Eng.* **29**(2): 47.
URL: *https://doi.org/10.1007/s10664-023-10433-5*

[7] ANYbotics [2025]. Anybotics - autonomous robotic inspection solutions, https://www.anybotics.com/. Accessed: 02.20.2025.

[8] Araujo, H., Mousavi, M. R. and Varshosaz, M. [2023]. Testing, validation, and verification of robotic and autonomous systems: A systematic review, *ACM Trans. Softw. Eng. Methodol.* **32**(2).
URL: *https://doi.org/10.1145/3542945*

[9] Ardupilot.org [2007]. Ardupilot – versatile, trusted, open, https://ardupilot.org/. accessed: 07.02.2022.

[10] Arnez, F., Espinoza, H., Radermacher, A. and Terrier, F. [2022]. Towards dependable autonomous systems based on bayesian deep learning components, *2022 18th European Dependable Computing Conference (EDCC)*, IEEE, pp. 65–72.

[11] Artzi, S., Kim, S. and Ernst, M. D. [2008]. Recrash: Making software failures reproducible by preserving object states, *in* J. Vitek (ed.), *Object-Oriented Programming, European Conference*, Vol. 5142 of *Lecture Notes in Computer Science*, Springer, pp. 542–565.
URL: *https://doi.org/10.1007/978-3-540-70592-5_23*

[12] Asmat, M. N., ur Rehman Khan, S. and Hussain, S. [2023]. Uncertainty handling in cyber-physical systems: State-of-the-art approaches, tools, causes, and future directions, *J. Softw. Evol. Process.* **35**(7).
URL: *https://doi.org/10.1002/smr.2428*

[13] Baumann, T. [2018]. Obstacle avoidance for drones using a 3dvfh* algorithm, *Spring Term 2018* **67**.

[14] Berndt, D. J. and Clifford, J. [1994]. Using dynamic time warping to find patterns in time series., *KDD workshop*, Vol. 10, Seattle, WA, USA:, pp. 359–370.

[15] Birchler, C., Ganz, N., Khatiri, S., Gambi, A. and Panichella, S. [2022]. Cost-effective simulation-based test selection in self-driving cars software with sdc-scissor, *the 29th IEEE International Conference on Software Analysis, Evolution, and Reengineering*.

[16] Birchler, C., Khatiri, S., Bosshard, B., Gambi, A. and Panichella, S. [2023]. Machine learning-based test selection for simulation-based testing of self-driving cars software, *Empir. Softw. Eng.* **28**(3): 71.
URL: *https://doi.org/10.1007/s10664-023-10286-y*

[17] Birchler, C., Khatiri, S., Rani, P., Kehrer, T. and Panichella, S. [2025]. A roadmap for simulation-based testing of autonomous cyber-physical systems: Challenges and future direction.
URL: *https://doi.org/10.1145/3711906*

[18] Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., Konolige, K. et al. [2018]. Using simulation and domain adaptation to improve efficiency of deep robotic grasping, *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, pp. 4243–4250.

[19] Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N. and Fox, D. [2019]. Closing the sim-to-real loop: Adapting simulation randomization with real world experience, *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 8973–8979.

[20] Cheng, C., Knoll, A. C. and Liao, H. [2021]. Safety metrics for semantic segmentation in autonomous driving, *2021 IEEE International Conference on Artificial Intelligence Testing, AITest 2021, Oxford, United Kingdom, August 23-26, 2021*, IEEE, pp. 57–64.
URL: *https://doi.org/10.1109/AITEST52744.2021.00021*

[21] Collins, J., Brown, R., Leitner, J. and Howard, D. [2020]. Traversing the reality gap via simulator tuning, *arXiv preprint arXiv:2003.01369* .

[22] Demir, S., Eniser, H. F. and Sen, A. [2020]. Deepsmartfuzzer: Reward guided test generation for deep learning, *Workshop on Artificial Intelligence Safety 2020 (IJCAI-PRICAI 2020)*, Vol. 2640 of *CEUR Workshop Proceedings*, CEUR-WS.org, pp. 134–140.

[23] Denney, E., Pai, G. and Habli, I. [2015]. Dynamic safety cases for through-life safety assurance, *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 2, IEEE, pp. 587–590.

[24] Di Sorbo, A., Zampetti, F., Visaggio, A., Di Penta, M. and Panichella, S. [2023]. Automated identification and qualitative characterization of safety concerns reported in uav software platforms, *ACM Trans. Softw. Eng. Methodol.* **32**(3).
URL: *https://doi.org/10.1145/3564821*

[25] Dimitropoulos, K., Hatzilygeroudis, I. and Chatzilygeroudis, K. [2022]. A brief survey of sim2real methods for robot learning, *International Conference on Robotics in Alpe-Adria Danube Region*, Springer, pp. 133–140.

[26] Dola, S., Dwyer, M. B. and Soffa, M. L. [2021]. Distribution-aware testing of neural networks using generative models, *International Conference on Software Engineering (ICSE)*, pp. 226–237.

[27] Dudzik, T., Chignoli, M., Bledt, G., Lim, B., Miller, A., Kim, D. and Kim, S. [2020]. Robust autonomous navigation of a small-scale quadruped robot in real-world environments, *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 3664–3671.

[28] Eiter, T. and Mannila, H. [1994]. Computing discrete fréchet distance, *Citeseer* .

[29] Galvan, J., Raja, A., Li, Y. and Yuan, J. [2021]. Sensor data-driven uav anomaly detection using deep learning approach, *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*, pp. 589–594.

[30] Gambi, A., Müller, M. and Fraser, G. [2019]. Automatically testing self-driving cars with search-based procedural content generation, *ACM SIGSOFT International Symposium on Software Testing and Analysis*, ACM, pp. 318–328.

[31] Gehring, C., Fankhauser, P., Isler, L., Diethelm, R., Bachmann, S., Potz, M., Gerstenberg, L. and Hutter, M. [2021a]. Anymal in the field: Solving industrial inspection of an offshore hvdc platform with a quadrupedal robot, *Field and Service Robotics: Results of the 12th International Conference*, Springer, pp. 247–260.

[32] Gehring, C., Fankhauser, P., Isler, L., Diethelm, R., Bachmann, S., Potz, M., Gerstenberg, L. and Hutter, M. [2021b]. ANYmal in the Field: Solving Industrial Inspection of an Offshore HVDC Platform with a Quadrupedal Robot, *Springer Proceedings in Advanced Robotics*, Vol. 16, Springer Science and Business Media B.V., pp. 247–260.

[33] Guiochet, J., Machin, M. and Waeselynck, H. [2017]. Safety-critical advanced robots: A survey, *Robotics and Autonomous Systems* **94**: 43–52.
URL: *https://www.sciencedirect.com/science/article/pii/S0921889016300768*

[34] Guo, J., Jiang, Y., Zhao, Y., Chen, Q. and Sun, J. [2018]. Dlfuzz: differential fuzzing testing of deep learning systems, *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ACM, pp. 739–743.
URL: *https://doi.org/10.1145/3236024.3264835*

[35] Ha, S., Lee, J., van de Panne, M., Xie, Z., Yu, W. and Khadiv, M. [0]. Learning-based legged locomotion: State of the art and future perspectives, *The International Journal of Robotics Research* **0**(0): 02783649241312698.
URL: *https://doi.org/10.1177/02783649241312698*

[36] Halder, S. and Afsari, K. [2023]. Robots in inspection and monitoring of buildings and infrastructure: A systematic review, *Applied Sciences* **13**(4).
URL: *https://www.mdpi.com/2076-3417/13/4/2304*

[37] Hao, W., Yang, T. and Yang, Q. [2023]. Hybrid statistical-machine learning for real-time anomaly detection in industrial cyber-physical systems, *IEEE Trans Autom. Sci. Eng.* **20**(1): 32–46.
URL: *https://doi.org/10.1109/TASE.2021.3073396*

[38] Hawkins, R., Habli, I., Kelly, T. and McDermid, J. [2013]. Assurance cases and prescriptive software safety certification: A comparative study, *Safety science* **59**: 55–71.

[39] Hildebrandt, C. and Elbaum, S. [2021]. World-in-the-loop simulation for autonomous systems validation, *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 10912–10919.

[40] Hoeller, D., Rudin, N., Sako, D. and Hutter, M. [2024]. Anymal parkour: Learning agile navigation for quadrupedal robots, *Science Robotics* **9**(88): eadi7566.

[41] Höfer, S., Bekris, K., Handa, A., Gamboa, J. C., Mozifian, M., Golemo, F., Atkeson, C., Fox, D., Goldberg, K., Leonard, J. et al. [2021]. Sim2real in robotics and automation: Applications and challenges, *IEEE transactions on automation science and engineering* **18**(2): 398–400.

[42] Huang, H., Savkin, A. V., Ding, M. and Huang, C. [2019]. Mobile robots in wireless sensor networks: A survey on tasks, *Computer Networks* **148**: 1–19.
URL: *https://www.sciencedirect.com/science/article/pii/S138912861830255X*

[43] Humbatova, N., Jahangirova, G., Bavota, G., Riccio, V., Stocco, A. and Tonella, P. [2020]. Taxonomy of real faults in deep learning systems, *in* G. Rothermel and D. Bae (eds), *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, ACM, pp. 1110–1121.
URL: *https://doi.org/10.1145/3377811.3380395*

[44] Humeniuk, D., Ben Braiek, H., Reid, T. and Khomh, F. [2024]. In-simulation testing of deep learning vision models in autonomous robotic manipulators, *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pp. 2187–2198.

[45] Humeniuk, D. and Khomh, F. [2024]. Ambiegen at the sbft 2024 tool competition - cps-uav track, *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024*.

[46] Hutter, M., Gehring, C., Lauber, A., Gunther, F., Bellicoso, C. D., Tsounis, V., Fankhauser, P., Diethelm, R., Bachmann, S., Blösch, M. et al. [2017]. Anymal-toward legged robots for harsh environments, *Advanced Robotics* **31**(17): 918–931.

[47] Ingrand, F. [2019]. Recent trends in formal validation and verification of autonomous robots software, *3rd IEEE International Conference on Robotic Computing, IRC 2019, Naples, Italy, February 25-27, 2019*, pp. 321–328.

[48] Jahangirova, G., Stocco, A. and Tonella, P. [2021]. Quality metrics and oracles for autonomous vehicles testing, *Conference on Software Testing, Verification and Validation*, IEEE, pp. 194–204.
URL: *https://doi.org/10.1109/ICST49551.2021.00030*

[49] James, S., Davison, A. J. and Johns, E. [2017]. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task, *Conference on Robot Learning*, PMLR, pp. 334–343.

[50] Javadi, A. and Birchler, C. [2025]. Tgen-uq at the icst 2025 tool competition - uav testing track, *IEEE/ACM International Conference on Software Testing, Verification and Validation, ICST 2025*. ICST Tool Competition 2025 - UAV Testing Track.

[51] Jiang, Z. and Babikian, A. A. [2025]. Optobstacles at the sbft 2025 tool competition - uav testing track, *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2025*.

[52] Kendall, A. and Gal, Y. [2017]. What uncertainties do we need in bayesian deep learning for computer vision?, *Advances in neural information processing systems* **30**.

[53] keras.io [2020]. Timeseries anomaly detection using an autoencoder. accessed: 03.04.2024.
URL: *https://keras.io/examples/timeseries/timeseries_anomaly_detection/*

[54] Khatiri, S., Amin, F. M., Panichella, S. and Tonella, P. [2025]. When uncertainty leads to unsafety: Empirical insights into the role of uncertainty in unmanned aerial vehicle safety, *arXiv preprint arXiv:2501.08908* .

[55] Khatiri, S., Barrientos, F. E. V., Wulf, M., Tonella, P. and Panichella, S. [2025]. Bridging research and practice in simulation-based testing of industrial robot navigation systems, *arXiv preprint arXiv:2510.09396* .

[56] Khatiri, S., Panichella, S. and Tonella, P. [2022]. replication package of the paper ”simulation-based test case generation for unmanned aerial vehicles in the neighborhood of real flight”.
URL: *https://doi.org/10.5281/zenodo.6525021*

[57] Khatiri, S., Panichella, S. and Tonella, P. [2023a]. Flight log for ”obstacle avoidance with a simple mission and a cargo container as obstacle”.
URL: *https://logs.px4.io/plot_app?log=f986a896-c189-4bfa-a11a-1d80fa4b9633*

[58] Khatiri, S., Panichella, S. and Tonella, P. [2023b]. Simulation-based test case generation for unmanned aerial vehicles in the neighborhood of real flights, *International Conference on Software Testing, Verification and Validation*, IEEE, pp. 281–292.
URL: *https://doi.org/10.1109/ICST57152.2023.00034*

[59] Khatiri, S., Panichella, S. and Tonella, P. [2024]. Simulation-based testing of unmanned aerial vehicles with aerialist, *46th IEEE/ACM International Conference on Software Engineering (ICSE), Lisbon, Portugal, 14-20 April 2024*.

[60] Khatiri, S., Saurabh, P., Zimmermann, T., Munasinghe, C., Birchler, C. and Panichella, S. [2024]. Sbft tool competition 2024: Cps-uav test case generation track, *17th International Workshop on Search-Based and Fuzz Testing (SBFT), Lisbon, Portugal, 14-20 April 2024*.

[61] Khatiri, S., Zohdinasab, T., Saurabh, P., Humeniuk, D. and Panichella, S. [2025a]. ICST tool competition 2025 - uav testing track, *IEEE/ACM International Conference on Software Testing, Verification and Validation, ICST 2025*.

[62] Khatiri, S., Zohdinasab, T., Saurabh, P., Humeniuk, D. and Panichella, S. [2025b]. SBFT tool competition 2025 - uav testing track, *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2025*.

[63] Kim, J., Feldt, R. and Yoo, S. [2019]. Guiding deep learning system testing using surprise adequacy, *International Conference on Software Engineering*, IEEE / ACM, pp. 1039–1049.
URL: *https://doi.org/10.1109/ICSE.2019.00108*

[64] Koenig, N. P. and Howard, A. [2004]. Design and use paradigms for gazebo, an open-source multi-robot simulator, *International Conference on Intelligent Robots and Systems*, IEEE, pp. 2149–2154.
URL: *https://doi.org/10.1109/IROS.2004.1389727*

[65] Koos, S., Mouret, J.-B. and Doncieux, S. [2012]. The transferability approach: Crossing the reality gap in evolutionary robotics, *IEEE Transactions on Evolutionary Computation* **17**(1): 122–145.

[66] Lechthaler, P., Prandi, D., Kifetew, F. M. and Susi, A. [2025]. Evolv-1 at the icst 2025 tool competition - uav testing track, *IEEE/ACM International Conference on Software Testing, Verification and Validation, ICST 2025*.

[67] Lee, T. E., Tremblay, J., To, T., Cheng, J., Mosier, T., Kroemer, O., Fox, D. and Birchfield, S. [2020]. Camera-to-robot pose estimation from a single image, *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 9426–9432.

[68] Lindvall, M., Porter, A., Magnusson, G. and Schulze, C. [2017]. Metamorphic model-based testing of autonomous systems, *International Workshop on Metamorphic Testing*, IEEE, pp. 35–41.

[69] Liso, M. D. and Soi, Z. W. [2024]. Camba cps-uav at the sbft tool competition 2024, *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024*.

[70] Liu, M., Xiao, J. and Li, Z. [2024]. Deployment of whole-body locomotion and manipulation algorithm based on nmpc onto unitree go2quadruped robot, *2024 6th International Conference on Industrial Artificial Intelligence (IAI)*, IEEE, pp. 1–6.

[71] Lu, H., Li, Y., Mu, S., Wang, D., Kim, H. and Serikawa, S. [2017]. Motor anomaly detection for unmanned aerial vehicles using reinforcement learning, *IEEE internet of things journal* **5**(4): 2315–2322.

[72] Luo, Q., Hariri, F., Eloussi, L. and Marinov, D. [2014]. An empirical analysis of flaky tests, *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, (FSE-22), Hong Kong, China, November 16 - 22, 2014*, pp. 643–653.

[73] Ma, L., Juefei-Xu, F., Xue, M., Li, B., Li, L., Liu, Y. and Zhao, J. [2019]. DeepCT: Tomographic combinatorial testing for deep learning systems, *International Conference on Software Analysis, Evolution and Reengineering*, IEEE, pp. 614–618.

[74] Ma, L., Juefei-Xu, F., Zhang, F., Sun, J., Xue, M., Li, B., Chen, C., Su, T., Li, L., Liu, Y., Zhao, J. and Wang, Y. [2018]. Deepgauge: Multi-granularity testing criteria for deep learning systems, *International Conference on Automated Software Engineering*, ACM, pp. 120–131.
URL: *http://doi.acm.org/10.1145/3238147.3238202*

[75] Ma, T., Ali, S. and Yue, T. [2021]. Testing self-healing cyber-physical systems under uncertainty with reinforcement learning: an empirical study, *Empir. Softw. Eng.* **26**(3): 52.
URL: *https://doi.org/10.1007/s10664-021-09941-z*

[76] Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A. et al. [2021]. Isaac gym: High performance gpu-based physics simulation for robot learning, *arXiv preprint arXiv:2108.10470* .

[77] Mankins, J. C. et al. [1995]. Technology readiness levels.

[78] Matternet [n.d.]. Faa grants approval to matternet for single pilot to operate up to 20 drones at test site, *Unmanned Publications* . accessed: 16.12.2023.

[79] McAllister, R. T., Gal, Y., Kendall, A., Van Der Wilk, M., Shah, A., Cipolla, R. and Weller, A. [2017]. Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning, *International Joint Conferences on Artificial Intelligence, Inc.*

[80] Meier, L., Honegger, D. and Pollefeys, M. [2015]. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms, *international conference on robotics and automation*, IEEE, pp. 6235–6240.

[81] Miki, T., Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V. and Hutter, M. [2022]. Learning robust perceptive locomotion for quadrupedal robots in the wild, *Science robotics* **7**(62): eabk2822.

[82] Muratore, F., Treede, F., Gienger, M. and Peters, J. [2018]. Domain randomization for simulation-based policy optimization with transferability assessment, *Conference on Robot Learning*, PMLR, pp. 700–713.

[83] Nahavandi, S., Alizadehsani, R., Nahavandi, D., Mohamed, S., Mohajer, N., Rokonuzzaman, M. and Hossain, I. [2025]. A comprehensive review on autonomous navigation, *ACM Comput. Surv.* **57**(9).
URL: *https://doi.org/10.1145/3727642*

[84] Narayanasamy, S., Pokam, G. and Calder, B. [2005]. Bugnet: Continuously recording program execution for deterministic replay debugging, *International Symposium on Computer Architecture*, IEEE Computer Society, pp. 284–295.
URL: *https://doi.org/10.1109/ISCA.2005.16*

[85] Ngo, A., Bauer, M. P. and Resch, M. [2021]. A multi-layered approach for measuring the simulation-to-reality gap of radar perception for autonomous driving, *24th IEEE International Intelligent Transportation Systems Conference, ITSC 2021, Indianapolis, IN, USA, September 19-22, 2021*, IEEE, pp. 4008–4014.
URL: *https://doi.org/10.1109/ITSC48978.2021.9564521*

[86] Pandey, A., Pandey, S. and Parhi, D. [2017]. Mobile robot navigation and obstacle avoidance techniques: A review, *Int Rob Auto J* **2**(3): 00022.

[87] Parra, S., Schneider, S. and Hochgeschwender, N. [2023]. A thousand worlds: scenery specification and generation for simulation-based testing of mobile robot navigation stacks, *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 5537–5544.

[88] Pei, K., Cao, Y., Yang, J. and Jana, S. [2019]. Deepxplore: Automated whitebox testing of deep learning systems, *Commun. ACM* **62**(11): 137?145.

[89] Petitjean, F., Ketterlin, A. and Gançarski, P. [2011]. A global averaging method for dynamic time warping, with applications to clustering, *Pattern recognition* **44**(3): 678–693.

[90] PX4 Developer Team [2022a]. Px4 autopilot user guide, https://docs.px4.io. accessed: 07.02.2022.

[91] PX4 Developer Team [2022b]. Px4 avoidance ros node for obstacle detection and avoidance, https://github.com/PX4/PX4-Avoidance. accessed: 07.02.2022.

[92] Riccio, V. and Tonella, P. [2020]. Model-based exploration of the frontier of behaviours for deep learning system testing, *ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 876–888.

[93] Ryan, P., Shahbeigi, S., Zou, J., Stefanakos, I. and Molloy, J. [2024]. A dynamic assurance framework for an autonomous survey drone, *International Conference on Computer Safety, Reliability, and Security*, Springer, pp. 285–299.

[94] Salvato, E., Fenu, G., Medvet, E. and Pellegrino, F. A. [2021]. Crossing the reality gap: a survey on sim-to-real transferability of robot controllers in reinforcement learning, *IEEE Access* .

[95] Shar, L. K., Minn, W., Ta, N. B. D., Fan, J., Jiang, L. and Kiat, D. L. W. [2022]. Dronlomaly: runtime detection of anomalous drone behaviors via log analysis and deep learning, *2022 29th Asia-Pacific Software Engineering Conference (APSEC)*, IEEE, pp. 119–128.

[96] Shrinah, A. and Eder, K. [2025]. Pseudo-random at the sbft 2025 tool competition - uav testing track, *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2025*.

[97] Sindhwani, V., Sidahmed, H., Choromanski, K. and Jones, B. [2020]. Unsupervised anomaly detection for self-flying delivery drones, *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 186–192.

[98] Solmaz, S., Innerwinkler, P., Wójcik, M., Tong, K., Politi, E., Dimitrakopoulos, G., Purucker, P., Höß, A., Schuller, B. W. and John, R. [2024]. Robust robotic search and rescue in harsh environments: An example and open challenges, *2024 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, IEEE, pp. 1–8.

[99] Soltani, M., Panichella, A. and van Deursen, A. [2020]. Search-based crash reproduction and its impact on debugging, *IEEE Trans. Software Eng.* **46**(12): 1294–1317.
URL: *https://doi.org/10.1109/TSE.2018.2877664*

[100] Sotiropoulos, T., Guiochet, G., Ingrand, I. and Waeselynck, W. [2016]. Virtual worlds for testing robot navigation: a study on the difficulty level, *2016 12th European Dependable Computing Conference (EDCC)*, IEEE, pp. 153–160.

[101] Sotiropoulos, T., Waeselynck, H., Guiochet, J. and Ingrand, F. [2017]. Can robot navigation bugs be found in simulation? an exploratory study, *2017 IEEE International conference on software quality, reliability and security (QRS)*, IEEE, pp. 150–159.

[102] Spahn, M., Salmi, C. and Alonso-Mora, J. [2022]. Local planner bench: Benchmarking for local motion planning, *arXiv preprint arXiv:2210.06033* .

[103] Stocco, A., Weiss, M., Calzana, M. and Tonella, P. [2020]. Misbehaviour prediction for autonomous driving systems, *in* G. Rothermel and D. Bae (eds), *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, ACM, pp. 359–371.
URL: *https://doi.org/10.1145/3377811.3380353*

[104] Tang, S., Zhang, Z., Cetinkaya, A. and Arcaini, P. [2024]. Tumb at the sbft 2024 tool competition – cps-uav test case generation track, *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024*.

[105] Tang, S., Zhang, Z., Cetinkaya, A. and Arcaini, P. [2025]. Palm at the icst 2025 tool competition - uav testing track, *IEEE/ACM International Conference on Software Testing, Verification and Validation, ICST 2025*. ICST Tool Competition 2025 - UAV Testing Track.

[106] Thill, M., Konen, W., Wang, H. and Bäck, T. [2021]. Temporal convolutional autoencoder for unsupervised anomaly detection in time series, *Applied Soft Computing* **112**: 107751.

[107] Tian, Y., Pei, K., Jana, S. and Ray, B. [2018]. Deeptest: Automated testing of deep-neural-network-driven autonomous cars, *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, ACM, pp. 303–314.
URL: *http://doi.acm.org/10.1145/3180155.3180220*

[108] Timperley, C. S., Afzal, A., Katz, D. S., Hernandez, J. M. and Le Goues, C. [2018]. Crashing simulated planes is cheap: Can simulation detect robotics bugs early?, *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, IEEE, pp. 331–342.

[109] Wang, D., Li, S., Xiao, G., Liu, Y. and Sui, Y. [2021]. An exploratory study of autopilot software bugs in unmanned aerial vehicles, *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 20–31.

[110] Winsten, J., Soloviev, V., Peltomäki, J. and Porres, I. [2024]. Adaptive test generation for unmanned aerial vehicles using wogan-uav, *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024*.

[111] Woodlief, T., Elbaum, S. and Sullivan, K. [2021]. Fuzzing mobile robot environments for fast automated crash detection, *International Conference on Robotics and Automation*, IEEE, pp. 5417–5423.

[112] Xie, X., Ma, L., Juefei-Xu, F., Xue, M., Chen, H., Liu, Y., Zhao, J., Li, B., Yin, J. and See, S. [2019]. Deephunter: A coverage-guided fuzz testing framework for deep neural networks, *ACM SIGSOFT International Symposium on Software Testing and Analysis*, Association for Computing Machinery, pp. 146–157.
URL: *https://doi.org/10.1145/3293882.3330579*

[113] Xu, Q., Ali, S., Yue, T. and Arratibel, M. [2022]. Uncertainty-aware transfer learning to evolve digital twins for industrial elevators, *in* A. Roychoudhury, C. Cadar and M. Kim (eds), *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, ACM, pp. 1257–1268.
URL: *https://doi.org/10.1145/3540250.3558957*

[114] Yan, C., Wang, N., Gao, H., Wang, X., Tang, C., Zhou, L., Li, Y. and Wang, Y. [2024]. An advanced reinforcement learning control method for quadruped robots in typical urban terrains, *International Journal of Machine Learning and Cybernetics* pp. 1–11.

[115] Yang, T., Hao, W., Yang, Q. and Wang, W. [2023]. Cloud-edge coordinated traffic anomaly detection for industrial cyber-physical systems, *Expert Syst. Appl.* **230**: 120668.
URL: *https://doi.org/10.1016/j.eswa.2023.120668*

[116] Yue, J. [2020]. Learning locomotion for legged robots based on reinforcement learning: A survey, *2020 International Conference on Electrical Engineering and Control Technologies (CEECT)*, IEEE, pp. 1–7.

[117] Zampetti, F., Kapur, R., Di Penta, M. and Panichella, S. [2022]. An empirical characterization of software bugs in open-source cyber–physical systems, *Journal of Systems and Software* **192**: 111425.
URL: *https://www.sciencedirect.com/science/article/pii/S0164121222001315*

[118] Zampetti, F., Tamburri, D. A., Panichella, S., Panichella, A., Canfora, G. and Penta, M. D. [2022]. Continuous integration and delivery practices for cyber-physical systems: An interview-based study, *ACM Trans. Softw. Eng. Methodol.* .
URL: *https://doi.org/10.1145/3571854*

[119] Zhang, F., Leitner, J., Ge, Z., Milford, M. and Corke, P. [2019]. Adversarial discriminative sim-to-real transfer of visuo-motor policies, *The International Journal of Robotics Research* **38**(10-11): 1229–1245.

[120] Zhu, T., Newton, W., Embury, S. and Sun, Y. [2024]. Taiist cps-uav at the sbft tool competition 2024, *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024*.

[121] Zohdinasab, T. and Doreste, A. [2024]. Deephyperion-uav at the sbft 2024 tool competition - cps-uav test case generation track, *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024*.

[122] Zohdinasab, T., Riccio, V., Gambi, A. and Tonella, P. [2023]. Efficient and effective feature space exploration for testing deep learning systems, *ACM Trans. Softw. Eng. Methodol.* **32**(2): 49:1–49:38.
URL: *https://doi.org/10.1145/3544792*