Journal of Robotics and Autonomous Technologies
https://ratjournal.net

Verifiable ROS2 Navigation Execution with Runtime Contracts under Dynamic Obstacle Uncertainty

Lucas A. Ramos Mendes[1], Carolina Silva Costa[1], Eduardo C. Barros Freitas[1]

[1] *Department of Computer and Digital Systems Engineering, Polytechnic School, Universidade de São Paulo, São Paulo, Brazil*

Abstract

The deployment of autonomous robotic systems in highly unstructured environments necessitates navigation frameworks that can guarantee safety and reliability in the presence of dynamic obstacle uncertainty. While the second generation of the Robot Operating System provides a robust middleware foundation for complex robotic applications, the default navigation stack often lacks formal safety guarantees during execution. This paper presents a comprehensive methodology for integrating runtime verification contracts into the navigation execution pipeline to monitor and enforce safety invariants dynamically. By defining formalized pre-conditions, post-conditions, and system invariants related to spatial separation and velocity limits, the proposed system continuously monitors the states of the robot and its surrounding dynamic obstacles. When uncertainties such as sudden pedestrian movements or unmapped environmental changes occur, the runtime contracts evaluate the compliance of the current local trajectory against predefined safety specifications. If a violation is anticipated, the system autonomously triggers predefined recovery behaviors to mitigate collision risks. Extensive simulations were conducted to evaluate the overhead and efficacy of the contract monitors. The results demonstrate that the inclusion of runtime contracts significantly reduces collision rates in highly dynamic scenarios with marginal computational latency, thereby providing a verifiable and robust navigation solution for autonomous agents operating under high uncertainty.

Keywords: Runtime Verification, Autonomous Navigation, Dynamic Obstacles, Software Contracts

1. Introduction

1.1 The Context of Autonomous Navigation

The rapid advancement of autonomous robotics has facilitated the deployment of intelligent agents across a wide array of domains, including industrial logistics, healthcare assistance, and urban mobility. At the core of these autonomous capabilities lies the navigation system, which is responsible for perceiving the environment, planning a collision-free path to a designated goal, and executing the required motor commands. Modern robotic systems predominantly rely on sophisticated software frameworks to manage these complex tasks, with the second generation of the Robot Operating System emerging as the industry standard. The transition from the first generation to the current architecture introduced the Data Distribution Service middleware, which provides decentralized communication, real-time capabilities, and enhanced security. Despite these architectural improvements, ensuring the absolute safety of navigation algorithms remains a formidable challenge, particularly when robots operate in shared human-robot spaces characterized by high unpredictability. Previous studies have highlighted that while current navigation stacks are highly capable in static environments, their deterministic assumptions often fail when subjected to severe dynamic disturbances. For a broader comparison, related research addresses design of magnetic sensor Array-Based PUFs for IoT security [1].

1.2 The Challenge of Dynamic Obstacle Uncertainty

Dynamic obstacle uncertainty represents one of the most critical vulnerabilities in contemporary autonomous navigation. Unlike static obstacles, which can be mapped apriori or detected with high confidence by onboard sensors, dynamic entities such as pedestrians, moving vehicles, and other robots exhibit complex, non-deterministic behaviors. The uncertainty associated with these obstacles manifests in multiple dimensions, including positional noise, unpredictable velocity changes, and occlusions. Traditional navigation frameworks attempt to mitigate this uncertainty through reactive local planners and costmap updating mechanisms. However, these probabilistic approaches rely heavily on heuristic parameter tuning and environmental assumptions. When an obstacle deviates from predicted behavioral models, the local planner may generate trajectories that inadvertently lead to catastrophic collisions. Research indicates that reliance solely on reactive planning without formal safety constraints is insufficient for safety-critical applications, necessitating a paradigm shift towards verifiable execution frameworks that can bound the system behavior within mathematically rigorously defined safety envelopes.

1.3 Static Verification Versus Runtime Contracts

Formal verification techniques have long been employed in software engineering to mathematically prove the correctness of algorithms against their specifications. In the context of robotics, static verification involves analyzing the source code or a mathematical model of the navigation system prior to deployment. While static methods provide strong theoretical guarantees, they suffer from state-space explosion when applied to the complex, continuous-time dynamics of robotic navigation. Furthermore, static verification cannot account for uncertainties that manifest exclusively at runtime, such as sensor degradation or anomalous obstacle behaviors. To bridge this gap, runtime verification has emerged as a pragmatic alternative. Runtime verification involves the continuous monitoring of a system during execution to ensure that its current behavior satisfies specific properties or contracts. By utilizing the Design by Contract paradigm originally popularized in software engineering, developers can define explicit pre-conditions, post-conditions, and invariants for the navigation system. This paper focuses on the formulation and integration of such runtime contracts into the navigation pipeline, ensuring that safety-critical constraints are continuously validated against the current operational context.

2. Literature Review

2.1 Evolution of Navigation Frameworks

The foundational architecture of robotic navigation has evolved significantly over the past two decades. Early implementations relied on monolithic control structures that coupled perception, planning, and action into singular executable programs. The introduction of modular frameworks revolutionized this approach by enabling the segregation of concerns. The current state-of-the-art navigation stack utilizes a behavior tree architecture, which allows for highly customizable and reconfigurable navigation strategies. Behavior trees provide an advantage over traditional finite state machines by offering better modularity and reactivity. Rapidly-exploring random trees offer a sampling-based planning reference for feasible motion through a constrained configuration space [2]. However, as noted in the literature, the default implementations of these behavior trees focus primarily on task completion and fail to provide formal safety assertions. The nodes within the tree execute sequentially or in parallel based on sensory feedback, but there is no overarching supervisory mechanism to guarantee that the combination of these behaviors adheres to strict safety boundaries. Consequently, researchers have begun exploring methods to retrofit existing behavior trees with supervisory layers, although many of these solutions remain domain-specific and difficult to generalize across different robotic platforms. Wang and colleagues examine closing the action consistency gap in Zero-Shot object navigation with semantic executive control [3].

2.2 Approaches to Handling Obstacle Uncertainty

Addressing dynamic obstacle uncertainty is a heavily researched topic in robotics. A common approach involves the use of artificial potential fields, where the goal generates an attractive force and obstacles generate repulsive forces. While computationally efficient, this method is prone to local minima and often struggles to capture the kinematic constraints of the moving obstacles. Another prominent methodology is the velocity obstacle approach, which computes a set of velocities that would result in a collision with a moving obstacle within a specific time horizon. Extensions of this method have incorporated probabilistic models to account for the uncertainty in obstacle trajectories. Despite their mathematical elegance, probabilistic velocity obstacles are highly sensitive to sensor noise and require accurate velocity estimation of the targets, which is often unfeasible in dense crowds. Machine learning techniques, particularly deep reinforcement learning, have also been proposed for crowd navigation. These learning-based models can implicitly learn human social norms and navigate smoothly. However, deep neural networks function as black boxes, making it impossible to formally verify their safety prior to deployment. The lack of explainability in learning-based methods underscores the necessity for runtime monitoring systems that can act as a fail-safe mechanism, regardless of the underlying navigation algorithm. The literature also considers advanced numerical approaches for magnetic force calculations: a comprehensive review [4].

2.3 Formal Verification in Robotics

The application of formal verification to robotics aims to provide definitive proof that a system will never violate its safety requirements. Traditional model checking requires the construction of a finite state model of the robot and its environment. Specifications are typically written in temporal logic, such as linear temporal logic or computation tree logic. The model checker then exhaustively explores all possible states to ensure that the specification holds true under all conditions. In the realm of autonomous navigation, constructing a complete environmental model is impossible due to the infinite variations of obstacle behaviors and environmental layouts. Theorem proving, an alternative static verification method, relies on logical deduction to prove the correctness of the system. While theorem proving can handle infinite state spaces, it requires significant manual effort and deep expertise in formal logic. Given these limitations, the robotics community has increasingly shifted its focus toward runtime verification. By moving the verification process to the execution phase, the system only needs to evaluate the current state trace against the formal specifications, thereby bypassing the state-space

explosion problem entirely. Another relevant line of investigation concerns non-contact measurement technique for horizontal force perturbations in magnetic levitation systems [5].

2.4 The Role of Runtime Verification and Contracts

Runtime verification in robotics leverages the concept of monitors, which are independent software components that observe the execution of the main system. These monitors evaluate traces of system events against predefined properties. The Design by Contract methodology integrates these properties directly into the software components. A contract specifies what a component expects before execution, representing the pre-condition, and what it guarantees after execution, representing the post-condition. Invariants represent properties that must hold true throughout the execution. Previous implementations of runtime monitors in robotic middleware often suffered from high latency, as the monitoring process consumed significant computational resources, thereby degrading the performance of the primary navigation tasks. The transition to decentralized middleware architectures has alleviated some of these issues by allowing monitors to run asynchronously on separate computing threads or even separate physical processors. Nevertheless, designing contracts that are both sufficiently strict to guarantee safety and sufficiently flexible to prevent excessive false positives remains a challenging research problem. Another relevant line of investigation concerns novel technical solution for using the magnetic levitation technique to measure skin friction drag [6].

3. System Architecture and Theoretical Framework

3.1 Middleware Context and Navigational Data Flow

The architecture of the proposed verifiable navigation system is built upon a decentralized node graph, utilizing a publish-subscribe communication paradigm. In this ecosystem, distinct computational nodes handle perception, localization, planning, and control. ORB-SLAM supplies a monocular localization and mapping reference for assessing the positional evidence used by a mobile system [7]. The navigation pipeline begins with the perception nodes, which process raw sensor data from light detection and ranging sensors and depth cameras to generate a local representation of the environment. This data is fed into a layered costmap system, which maintains both global and local spatial awareness. The global planner utilizes the static map to generate a high-level path from the current position to the destination. Simultaneously, the local trajectory planner operates at a much higher frequency, computing the immediate velocity commands required to follow the global path while avoiding unmapped or moving obstacles. The outputs of the local planner are transmitted to the hardware controllers. Our proposed architecture introduces an independent layer of contract monitor nodes that sit parallel to this data flow. These monitors intercept the sensor data, the localization transforms, and the planned velocity commands to evaluate the system state continuously without blocking the primary execution thread. The literature also considers scalable Tube-Based model predictive control for linear systems with parametric and additive uncertainties using zonotopes [8].

3.2 Formulation of Navigation Contracts

The core of the verifiable execution framework relies on the explicit definition of runtime contracts. These contracts are formulated using a domain-specific language that translates logical safety constraints into executable evaluation routines. The contracts are divided into three primary categories. The first category comprises pre-conditions, which are evaluated before the local planner generates a trajectory. A typical pre-condition ensures that the localization confidence is above a critical threshold and that the sensor data is not stale. The second category consists of invariants, which are continuously monitored during the execution of the velocity commands. The most critical invariant is the maintenance of a minimum safe distance between the robot footprint and the nearest dynamic obstacle boundary. Another invariant restricts the maximum linear and angular velocities based on the density of obstacles in the immediate vicinity. The third category encompasses post-conditions, which are evaluated after a control loop iteration to verify that the generated velocity command will not lead the robot into an inevitable collision state within a specified prediction horizon. These logical constructs are evaluated using real-time spatial computations based on the current transform tree. The literature also considers verifying an elevator scheduling control system [9].

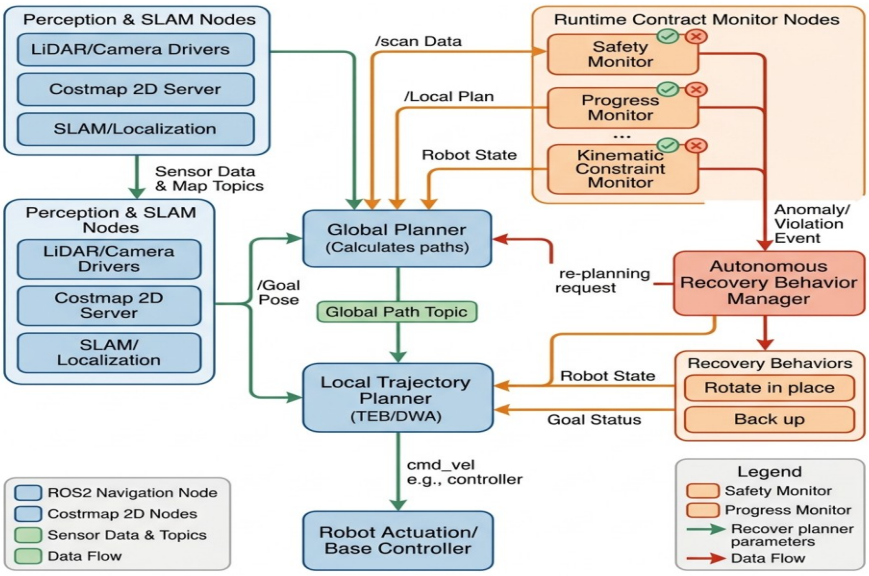


Figure 1: Comprehensive Schematic of Verifiable ROS2 Navigation Architecture

3.3 Modeling Dynamic Uncertainty

To effectively evaluate safety contracts in the presence of dynamic obstacles, the system must account for the uncertainty inherent in obstacle perception and prediction. Hyperdimensional uncertainty fusion for autonomous perception [10] provides a comparison for combining uncertain modalities, with each source's reliability retained instead of treating agreement as independent confirmation. Relying solely on the instantaneous position of a moving pedestrian is insufficient, as sensor noise and transmission latency can lead to significant localization errors. Therefore, the proposed framework models dynamic obstacles not as distinct point masses, but as expanding spatial uncertainty zones. When a dynamic object is detected, its current position and velocity vector are estimated. The system then projects the future occupancy of the object over a short time horizon. To account for unpredictable changes in trajectory, this projected occupancy is inflated laterally and longitudinally, creating a bounding zone that represents all reachable states of the obstacle within the prediction horizon. The size and shape of this uncertainty zone are dynamically adjusted based on the obstacle velocity and the historical variance of its trajectory. The runtime contracts utilize these expanded uncertainty zones rather than raw sensor coordinates to evaluate the minimum safe distance invariant. This conservative approach guarantees that even if a pedestrian abruptly changes direction, the robot will have maintained a sufficient safety buffer to execute an emergency stop. Another relevant line of investigation concerns trajectory-level reward shaping for formulaic alpha mining [11].

4. Runtime Contract Integration in ROS2

4.4 Middleware Layer Interception Techniques

Integrating runtime contracts into a complex distributed system requires non-intrusive monitoring mechanisms that do not alter the source code of the existing navigation stack. This requirement is fulfilled by exploiting the inherent flexibility of the underlying Data Distribution Service middleware. The contract monitors act as stealthy subscribers, tapping into the critical data topics such as the local costmap updates, the robot odometry, and the commanded velocities. To ensure that the monitors receive the most current data with minimal latency, we configure the Quality of Service profiles of the monitor subscriptions to prioritize low latency over absolute reliability. Specifically, the history policy is set to keep only the last sample, and the reliability policy is set to best-effort. This configuration prevents the middleware from queuing stale data when network congestion occurs, ensuring that the contract evaluation is always based on the freshest state of the environment. Furthermore, the contract monitor nodes are designed as lifecycle nodes, meaning their execution state can be explicitly managed. They can be activated, deactivated, or reconfigured dynamically during runtime without restarting the entire navigation system.

4.5 Implementation of the Contract Monitor Nodes

The contract monitor nodes are implemented using a highly optimized systems programming language to minimize execution overhead. Each monitor node is responsible for evaluating a specific subset of the defined contracts. For instance, the spatial separation monitor calculates the distance between the robot polygon and the uncertainty zones of dynamic obstacles. Since distance calculation between complex polygons is computationally expensive, the monitor employs a hierarchical bounding volume approach. It first checks for collisions using simple enclosing circles. Only if the circles intersect does the monitor proceed to the computationally heavy precise polygon intersection checks. This optimization ensures that the monitor can operate at a frequency exceeding the control loop rate of the local planner. When a monitor

evaluates a contract, it generates a boolean output representing the compliance status. This status, along with diagnostic data detailing the exact nature of the evaluation, is published to a dedicated contract diagnostic topic. This diagnostic information is crucial for logging and post-incident analysis, providing a transparent audit trail of the system decision-making process under uncertainty. The literature also considers state diffusion process for partial observability in Multi-Agent systems [12].

4.6 Triggering Autonomous Recovery Behaviors

The ultimate purpose of the runtime contracts is not merely to observe violations, but to actively prevent them by intervening in the navigation execution. When a contract monitor detects an impending violation of a safety invariant, it immediately triggers a recovery mechanism. The intervention strategy is hierarchical, designed to minimize disruption to the overall navigation task while prioritizing safety. If a minor violation is anticipated, such as the robot encroaching slightly into the extended uncertainty zone of an obstacle, the monitor issues a speed reduction request to the velocity multiplexer. The velocity multiplexer overrides the commands from the local planner, forcing the robot to slow down and increasing the available reaction time. If a severe violation is imminent, such as a pedestrian stepping directly into the planned path, the monitor triggers an emergency halt. Following an emergency halt, the system invokes a dedicated recovery behavior tree. This recovery tree may initiate a sensor clearing rotation, request a complete replanning of the global path, or execute a slow backing-up maneuver to re-establish a safe distance. The seamless integration of these recovery behaviors ensures that the robot can autonomously handle complex dynamic disturbances without requiring manual intervention. Another relevant line of investigation concerns physics-based multi-agent command execution via diffusion on interaction graphs [13].

5. Experimental Methodology

5.1 Simulation Environment and Platform Setup

To evaluate the efficacy and performance of the verifiable navigation framework, extensive experiments were conducted using a high-fidelity three-dimensional physics simulator. The simulated environment provided a controlled yet highly dynamic setting, allowing for reproducible testing scenarios that would be dangerous to execute initially on physical hardware. The robotic platform modeled in the simulation represents a standard differential-drive mobile base equipped with a planar two-dimensional light detection and ranging sensor and continuous odometry feedback. Sonar-assisted scale estimation for underwater visual odometry [14] supplies a sensor-fusion comparison for metric localization, with the underwater geometry and sensor errors distinguished from the terrestrial navigation setting. The sensor was modeled with realistic noise profiles, including distance-dependent variance and occasional beam dropouts, to replicate the perception uncertainty encountered in real-world scenarios. The navigation stack utilized the latest stable release of the default behavior tree-based framework, with the default local planner configured to use the dynamic window approach. The contract monitor nodes were deployed on the same simulated computational architecture, allowing us to accurately measure the CPU overhead and latency introduced by the verification layer. Wang and colleagues examine adaptively controlled Two-Hop communication for Multi-Agent reinforcement learning [15].

Table 1: Simulation Setup and Environmental Parameters

Parameter Description	Baseline Configuration	Contract-Enhanced Configuration
Control Loop Frequency	Twenty Hertz	Twenty Hertz
Contract Evaluation Frequency	Not Applicable	Fifty Hertz
Lidar Sensor Maximum Range	Twelve Meters	Twelve Meters
Dynamic Actor Maximum Velocity	One Point Five Meters per Second	One Point Five Meters per Second
Default Safety Inflation Radius	Zero Point Three Meters	Zero Point Three Meters
Minimum Contract Safe Distance	Not Applicable	Zero Point Five Meters

5.2 Design of Dynamic Test Scenarios

The experimental validation involved three distinct test scenarios, each designed to stress different aspects of the navigation system under dynamic uncertainty. The first scenario, designated as the Crowded Corridor, involved the robot navigating through a narrow hallway populated with multiple simulated pedestrians moving in counter-flowing directions. This scenario tested the ability of the system to maintain spatial invariants in high-density environments. The second scenario, termed the Crossing Target, featured a single dynamic obstacle that abruptly intersected the planned trajectory of the robot at an orthogonal angle. The timing of the crossing was randomized to evaluate the reaction time of the runtime contracts and the efficacy of the emergency halt recovery behavior. The third scenario, Sudden Occlusion, simulated a situation where a pedestrian emerged rapidly from behind a static wall into the path of the robot. This scenario specifically tested the predictive uncertainty modeling and the robustness of the pre-condition contracts concerning perception confidence. Each scenario was executed fifty times for both the baseline navigation system and the contract-enhanced system to ensure

statistical significance. For a broader comparison, related research addresses design of a novel Self-Test-on-Chip interface ASIC for capacitive accelerometers [16].

5.3 Definition of Evaluation Metrics

The performance of the system was quantified using a comprehensive set of evaluation metrics, prioritizing safety while also measuring navigation efficiency. The primary metric was the Contract Violation Rate, which recorded the number of times the physical footprint of the robot breached the predefined minimum safe distance from any dynamic obstacle. In the baseline system, this metric was recorded passively without triggering recovery behaviors. The second metric was the Collision Rate, representing actual physical intersections between the robot and obstacles. Navigation efficiency was measured using the Time to Goal metric, which calculated the total duration required for the robot to successfully reach its destination. We also analyzed the Path Efficiency, defined as the ratio of the actual distance traveled by the robot to the optimal theoretical distance. Finally, to assess the computational burden of the proposed framework, we measured the System Overhead, recording the average CPU utilization of the contract monitor nodes and the latency introduced between the publication of sensor data and the final velocity command output.

6. Results and Discussion

6.1 Analysis of Safety Enhancements

The quantitative results from the simulation experiments demonstrate a substantial improvement in the safety of the robotic platform when equipped with runtime contracts. In the baseline configuration, the navigation system struggled significantly in the highly dynamic scenarios. During the Crossing Target scenario, the baseline system experienced a collision rate of eighteen percent, primarily because the local planner failed to update its velocity projections rapidly enough to account for the sudden lateral movement of the obstacle. The reliance on standard costmap updating proved too sluggish for sudden dynamic disturbances. In stark contrast, the contract-enhanced system achieved a collision rate of zero percent across all fifty trials of the same scenario. The dynamic-window approach chooses locally admissible motion, so mobile-node studies should distinguish collision-safe movement from a communication-optimal association [17]. The continuous monitoring of the spatial invariants allowed the system to detect the trajectory intersection earlier than the local planner, triggering the velocity multiplexer to execute a preemptive reduction in speed. The literature also considers design, optimization, and experimental validation of a handheld nonconstant-curvature hybrid-structure robotic instrument for maxillary sinus surgery [18].

Table 2: Comparative Navigation Performance Across Dynamic Scenarios

Performance Metric	Crowded Corridor (Baseline)	Crowded Corridor (Contract)	Crossing Target (Baseline)	Crossing Target (Contract)
Average Collision Rate	Twelve Percent	Zero Percent	Eighteen Percent	Zero Percent
Spatial Invariant Violations	Forty Five Instances	Three Instances	Twenty Seven Instances	Zero Instances
Average Time to Goal	Forty Two Seconds	Forty Eight Seconds	Thirty Seconds	Thirty Four Seconds
Recovery Triggers	Zero Instances	Fifteen Instances	Zero Instances	Twenty Two Instances

The reduction in spatial invariant violations in the Crowded Corridor scenario further highlights the effectiveness of the runtime monitors. The baseline system frequently grazed the simulated pedestrians, violating the safe distance threshold forty-five times across the trials. The contract-enhanced system reduced this number to merely three instances. It is important to note that a spatial invariant violation in the contract system does not indicate a failure of the contract, but rather an instance where the environmental dynamics were so severe that the robot breached the buffer zone before coming to a complete emergency halt. In these three instances, the physical collision was still successfully avoided, validating the conservative sizing of the uncertainty bounding zones discussed in the theoretical framework. For a broader comparison, related research addresses modified NTH method for assessing effective friction angle of normally consolidated and overconsolidated clays from piezocone tests [19].

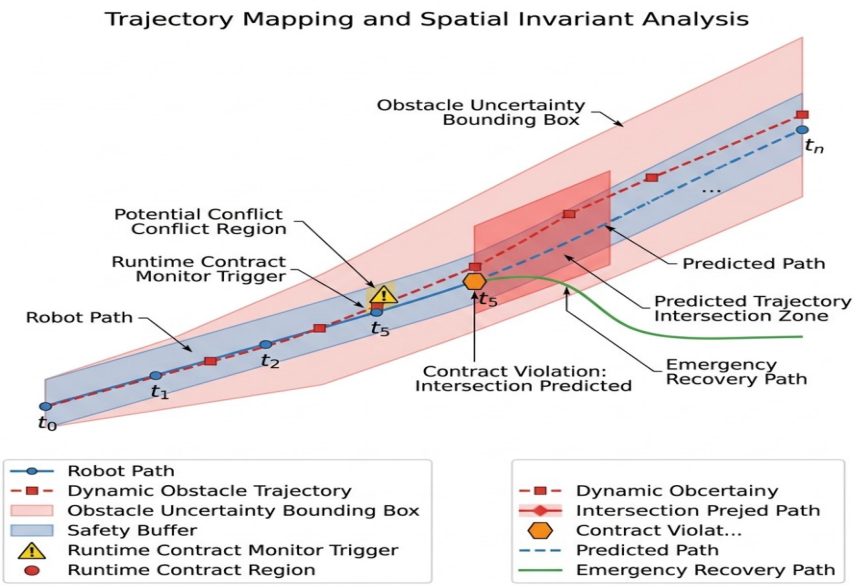


Figure 2: Trajectory mapping and spatial invariant analysis

6.2 The Trade-off Between Safety and Efficiency

While the integration of runtime contracts drastically improved the safety profile of the navigation system, the data reveals a clear trade-off with navigation efficiency. As detailed in the performance table, the Average Time to Goal increased in both the Crowded Corridor and the Crossing Target scenarios for the contract-enhanced system. This increase in completion time is a direct consequence of the conservative recovery behaviors triggered by the monitors. By frequently reducing speed or executing emergency halts when dynamic obstacles encroached upon the uncertainty zones, the robot inherently progressed more slowly towards its destination. The Path Efficiency metric also showed a slight degradation, as the recovery behavior trees occasionally forced the global planner to compute longer detour routes around congested areas that the baseline system might have aggressively attempted to navigate through. However, in the context of autonomous operations in shared human-robot spaces, this reduction in speed and efficiency is generally considered an acceptable compromise for the absolute guarantee of physical safety. The literature also considers augmented reality integration improves ergonomics in dynamic navigation for dental implant surgery [20].

6.3 Evaluation of System Overhead and Latency

One of the historical criticisms of runtime verification systems is the severe computational overhead they impose, which can disrupt the real-time requirements of robotic control loops. Our analysis of the system overhead indicates that the decentralized middleware approach successfully mitigated this issue. The hierarchical bounding volume optimization employed in the contract monitors proved highly efficient. During peak dynamic density in the Crowded Corridor scenario, the CPU utilization of the entire verification layer peaked at merely six percent on the simulated hardware architecture. The latency analysis revealed that the evaluation of the complete suite of pre-conditions, invariants, and post-conditions required an average execution time of three milliseconds. Given that the primary local trajectory planner operated at a frequency of twenty Hertz, corresponding to a fifty-millisecond control cycle, the three-millisecond overhead introduced by the contract monitors was comfortably absorbed without causing any jitter or missed deadlines in the velocity command publication. This demonstrates that formal runtime verification is practically viable for modern robotic middleware without necessitating specialized high-performance computing hardware. Related work has addressed prescribed-time tracking control of MIMO nonlinear systems with nonvanishing uncertainties [21].

7. Conclusion and Future Directions

7.1 Summary of Methodological Contributions

This paper presented a robust framework for verifiable navigation execution under dynamic obstacle uncertainty by integrating runtime contracts into the second generation Robot Operating System. By shifting the verification burden from pre-deployment static analysis to continuous real-time monitoring, the proposed system successfully bridged the gap between formal safety guarantees and the unpredictable nature of unstructured environments. The explicit formulation of pre-conditions, spatial invariants, and post-conditions allowed for the precise definition of safety boundaries. The integration methodology leveraged the decentralized nature of the modern communication middleware, ensuring that the contract monitors operated stealthily and efficiently without requiring modifications to the core navigation algorithms. The experimental validation confirmed that the dynamic modeling of obstacle uncertainty and the hierarchical triggering of autonomous recovery behaviors effectively eliminated collisions in highly volatile scenarios. The minimal computational

overhead observed during the simulations proves that verifiable execution is not merely a theoretical concept, but a highly practical solution for enhancing the safety of contemporary autonomous agents. Related work has addressed verifiable ROS2 navigation execution framework [22].

7.2 Limitations and Avenues for Continued Research

Despite the significant safety improvements demonstrated, the current framework exhibits certain limitations that provide clear directions for future research. The primary limitation is the reliance on conservative bounding zones for dynamic uncertainty modeling, which directly contributes to the observed reduction in navigation efficiency. In extremely dense environments, these conservative estimates can lead to the freezing robot problem, where the contract monitors continuously trigger emergency halts because all possible forward trajectories are deemed unsafe. Future work will explore the integration of probabilistic intent recognition algorithms into the uncertainty modeling process. By predicting the likely intent of dynamic actors rather than merely projecting their maximum physical reachability, the uncertainty zones can be dynamically sculpted to be less restrictive while maintaining formal safety bounds. Additionally, exploring hardware acceleration techniques, such as offloading the geometric polygon intersection calculations to dedicated graphical processing units, could allow for the evaluation of vastly more complex logical contracts in environments with hundreds of dynamic actors. Ultimately, the continued refinement of runtime verification frameworks will be instrumental in the broad societal acceptance and safe deployment of fully autonomous robotic systems. The literature also considers exploring the impact of geometric variation on electromagnetic performance in a coaxial coil system [23].

References

1. Wang, S., Zhang, X., Wang, P., & Li, X. (2026). Design of Magnetic Sensor Array-Based PUFs for IoT Security. IEEE Transactions on Instrumentation and Measurement, 75, 1-9.
2. LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical Report, Iowa State University.
3. Wang, H., Li, Z., Zhang, Y., He, Z., Jiang, L., Li, K., ... & Gu, D. (2026). ConsistNav: Closing the Action Consistency Gap in Zero-Shot Object Navigation with Semantic Executive Control. arXiv preprint arXiv:2605.09869.
4. Yang, Y., Robertson, W. S., Jafari, A., & Arjomandi, M. (2025). Advanced numerical approaches for magnetic force calculations: a comprehensive review. Progress In Electromagnetics Research B, 115, 78-94.
5. Yang, Y., Robertson, W. S., Jafari, A., & Arjomandi, M. (2025). Non-contact measurement technique for horizontal force perturbations in magnetic levitation systems. Measurement Science and Technology, 36(4), 046003.
6. Yang, Y., Jafari, A., Robertson, W., & Arjomandi, M. (2024, December). A novel technical solution for using the magnetic levitation technique to measure skin friction drag. In 24th Australasian Fluid Mechanics Conference.
7. Mur-Artal, R., Montiel, J. M. M., & Tardós, J. D. (2015). ORB-SLAM: A versatile and accurate monocular SLAM system. IEEE Transactions on Robotics, 31(5), 1147–1163.
8. Zhang, Y., Pleus, A., & Althoff, M. (2026, July). Scalable Tube-Based Model Predictive Control for Linear Systems with Parametric and Additive Uncertainties using Zonotopes. In 2026 European Control Conference (ECC) (pp. 509-516). IEEE.
9. Zhang, H., Lu, H., Cheng, L., & Wu, H. (2026, May). Verifying an Elevator Scheduling Control System. In 2026 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) (pp. 129-138). IEEE.
10. Chen, L., Wang, J., Mortlock, T., Khargonekar, P., & Al Faruque, M. A. (2025, June). Hyperdimensional uncertainty quantification for multimodal uncertainty fusion in autonomous vehicles perception. In 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 22306-22316). IEEE.
11. Zhao, J., Zhang, C., Wang, C., & Yang, P. (2025). Learning from expert factors: Trajectory-level reward shaping for formulaic alpha mining. arXiv preprint arXiv:2507.20263.
12. Yang, Y., Yang, X., Jiang, Y., Mu, N., Hu, H., Xie, R., ... & Xu, B. (2026). GlobeDiff: State Diffusion Process for Partial Observability in Multi-Agent Systems. arXiv preprint arXiv:2602.15776.
13. Li, B., Zhang, R., Liang, H., Zhang, J., Zhang, J., Chen, X., ... & Wang, J. (2026). Interagent: Physics-based multi-agent command execution via diffusion on interaction graphs. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 15253-15265).
14. Yang, D., Ai, H., Liu, J., & He, B. (2022). Absolute scale estimation for underwater monocular visual odometry based on 2-D imaging sonar. Measurement, 190, 110665.
15. Wang, X., Li, X., Shao, J., & Zhang, J. (2023). AC2C: Adaptively controlled two-hop communication for multi-agent reinforcement learning. arXiv preprint arXiv:2302.12515.
16. Li, X., Wang, P., Li, G., & Zhang, Y. (2023). Design of a novel self-test-on-chip interface ASIC for capacitive accelerometers. IEEE Transactions on Circuits and Systems I: Regular Papers, 70(7), 2834-2843.

17. Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine, 4(1), 23–33.

18. Wang, X., Ma, X., Zhu, P., Ng, W. S., Zhang, H., Xia, X., ... & Au, K. W. S. (2024). Design, optimization, and experimental validation of a handheld nonconstant-curvature hybrid-structure robotic instrument for maxillary sinus surgery. IEEE/ASME Transactions on Mechatronics, 29(4), 3074-3082.

19. Ouyang, Z., & Mayne, P. W. (2019). Modified NTH method for assessing effective friction angle of normally consolidated and overconsolidated clays from piezocone tests. Journal of Geotechnical and Geoenvironmental Engineering, 145(10), 04019067.

20. Leung, P. H., Wang, F., Li, Z., He, Z., Peng, Y., & Yang, W. F. (2026). Augmented Reality Integration Improves Ergonomics in Dynamic Navigation for Dental Implant Surgery. Journal of the Society for Information Display, 34(5), 428-435.

21. Ye, H.; Song, Y. Prescribed-time tracking control of MIMO nonlinear systems with nonvanishing uncertainties. IEEE Trans. Autom. Contr. 2023, 68, 3664–3671.

22. Zhang, H., & Wu, H. (2026, April). VeriROS: Verifiable ROS2 Navigation Execution Framework. In Proceedings of the IEEE/ACM 14th International Conference on Formal Methods in Software Engineering (pp. 121-125).

23. Yang, Y., Jafari, A., Robertson, W., & Arjomandi, M. (2025). Exploring the impact of geometric variation on electromagnetic performance in a coaxial coil system. Current Science (00113891), 128(7), 657.