

RUNTIME VERIFICATION AND PREDICTIVE SAFETY SHIELDS FOR AUTONOMOUS AGENTS: A GRIDWORLD SIMULATION STUDY

Gaurav Chaulagain¹, Medhavee Upadhyaya²

^{1,2} Department of Electrical and Computer Engineering, San Francisco Bay University

Abstract: Autonomous agents perform in variable conditions where acquired or heuristic decision-making procedures could engage in actions that contradict safety guidelines. While safety shields can react before a potentially harmful action is executed, runtime verification finds differences from formal requirements. However, when these factors are combined, it leads to a trade-off between safety and utility due to inaccurate observations, unclear barrier dynamics, and limited time for decision-making. Modules such as a three-valued temporal monitor, a finite-horizon action risk filter, a minimum-modeled-risk contingency plan, and an observation-focused probabilistic motion prediction system are presented in this paper as Runtime Verification with Predictive Shielding (RV-PS). In a controlled setting, there are 240 navigational realms evaluated at four levels of sensor mislocalization; in five simplified controllers, there are 960 shared environments and 4,800 paired policy implementations. In comparison to the unshielded

control (24.17%) and traditional predictive shielding (16.98%), the proposed RV-PS shows an average collision-episode rate of 2.08% across several scenarios at a three-step interval and a modeled-risk threshold of 0.075. As an example of the consequences of being too careful, its mean task completion rate is 71.35%, which is lower than the conventional shield's 82.50%. Completion rates are increased from 73.75% to 88.75% with a relaxed threshold of 0.20 and collision rates are raised from 2.92% to 4.17% when there is 25% sensor noise. Ablations expose vulnerabilities that can be addressed through anticipatory actions, dealing with uncertainty, and choosing backup plans. Under any given sensor-noise scenario, all policy variants use the identical generated obstacle pathways. In addition to a thorough analysis of safety, efficacy, and any remaining shortcomings, the research offers concrete evidence at the level of specifications

Key words: Runtime verification, Safety shields, Autonomous agents, Safe reinforcement learning, Temporal logic, Risk-aware control, Gridworld simulation.

Introduction

1.1 Research context

Autonomous agents make decisions on what to do based on factors such as internal rules, learned models, sensor data, and changing environmental conditions. There are a variety of uses for mobile robots, including automated services, unmanned vehicles, and agent cooperation. Because these systems are self-governing, there is a unique assurance problem: an agent can achieve traditional

goals while also acting in a way that benefits them locally, even though this goes against a fundamental safety need. Observable execution sequences are examined during runtime verification in cyber-physical contexts, when formal constraints must be met while dealing with sensors, timing, and physical ambiguity.

An important but distinct matter is addressed by a safety shield. It examines a proposal for action and may cancel it before the environment transition happens, rather than just saying that a trace broke a property. How this intervention can prevent disruption while imposing model-defined constraints has been investigated in research on quantitative games and multi-agent shield synthesis. Research on probabilistic and deterministic shielding explains why uncertainty-aware constraints are different from blanket policies.

1.2 Motivation and problem statement

In the next interval, two robots may enter the same empty cell, causing the first robot to crash into the second. Therefore, guaranteeing motion safety requires more than just analyzing the current observation. Conversely, time-sensitive assignments may go unfulfilled due to unneeded delays, diversions, or a barrier that kills all cells adjacent to a potentially moveable obstruction. Although projections raise expectations, they are based on

inaccurate observations and basic motion models.

Consequently, this study investigates the possibility of combining a runtime monitor with a predictive shield to make sure that safety judgments can be independently checked, that approximation can be traced back to its source, and that performance costs can be measured. The proposed architecture is targeted for navigation agents with finite states.

1.3 Objectives and research questions

The goals are to describe a framework for monitor-shield execution, build a replicable proof-of-concept simulator, test shield policies under common stochastic perturbations, study uncertainty sensitivity, and find the components that cause variations in simulated collision and task completion rates.

Three research questions guide the study:

RQ1: What is the impact of predictive runtime shield on collision frequency compared to unshielded and reactive strategies?

RQ2: What utility and intervention challenges accompany more stringent regulations?

RQ3: What effect do different tactics, the prediction horizon, risk restrictions, and inaccurate sensors have on the results that are simulated?

1.4 Contributions and scope

This paper expands upon previous work by offering a three-stage observation-based risk filter with a traceable intervention rule, a paired simulation with episode-level outputs, a conditional safety analysis that distinguishes between safe set claims and probabilities, and a clear separation of agent authority from monitor and shield authority.

2. Related Work

2.1 Runtime Verification and Reactive Safety Shields

During runtime verification, the monitoring logic is extracted from formal specifications and the execution prefix that an operational system generates is evaluated. New research highlights the challenges of transferring runtime monitoring approaches from traditional software environments to real-world, distributed, and unpredictable systems [1]. One way to think about optimizing learning controller runtime is as a quantitative game, where safety shields measure intervention costs against predetermined performance targets [2]. Protective techniques for multi-agent systems that are both efficient and fair have been investigated; in such systems, local changes to actions must be in line with global safety rules [3].

Agents can now incorporate safety limitations into their action choices through the implementation of probabilistic shielding in reinforcement learning within Markov decision processes [4]. Furthermore, several safety enforcement approaches have been evaluated using deterministic, probabilistic, and temporal shielding methodologies in reinforcement learning situations [5]. This research shows that runtime mediation is crucial for preventing dangerous autonomous behaviors. However, the accuracy and reliability of the underlying abstract system models are crucial to these guaranties. The insignificance of observational noise, environmental unpredictability, and unanticipated physical occurrences calls for focused attention.

2.2 Safe Learning, Barrier Functions, and Constrained Optimization

Mathematical safety restrictions have been increasingly implemented into autonomous controllers with learning ability to reduce unsafe behavior. Acquired control techniques can function

within predetermined safety limits, according to methods that integrate machine learning, Gaussian-process uncertainty assessment, and control barrier functions [6]. Learning-augmented control systems might benefit from reachability-oriented safety frameworks, which provide additional ways to limit potentially dangerous actions [7]. On the other hand, studies have looked into learning dynamics corrections to improve safety-critical control when dealing with imperfect system models [9], and model uncertainty is integrated within the constraints of control barriers and control Lyapunov functions [8]. Although these approaches often need for more complex mathematical formulations and more processing power, they provide stronger physical-control foundations than finite-state safety models.

Another way to include safety requirements into autonomous decision-making is with restricted reinforcement learning. In order to make reinforcement learning's constraint management more stable, PID Lagrangian methods have been developed. Encouraging agents to stay within confirmed boundaries is the goal of techniques for secure exploration in constrained Markov decision systems [11]. While optimization methods that focus on projection adjust updated rules so they conform to distributions of allowable actions [12], optimization methods that employ penalty systems to encourage compliance with restrictions [13] are known as reward-constrained optimization. An structured set of criteria for evaluating safe reinforcement learning approaches is provided by synthetic evaluation frameworks such as Safety Gym [14]. But results from simulations can't guaranty operational safety in the actual world on their own. The proposed RV-PS system uses a static heuristic policy to do runtime safety filtering, in contrast to current training-focused techniques. Because of the limitations of existing reinforcement learning algorithms, this study does not claim to have achieved direct experimental dominance.

2.3 Formal Analysis, Scenario Assessment, and Learned Controllers

The effectiveness of autonomous systems with learning capabilities can be evaluated through the use of formal verification and simulation-driven testing. In order to identify potential safety violations, simulation-oriented falsification frameworks combine explicit requirements with systematic scenario building [15]. Methods for evaluating the safety of autonomous vehicles that are scenario-oriented call attention to the limitations of simulation comprehensiveness and the necessity of testing in actual situations [16]. According to extensive research on autonomous driving frameworks, uncertainty in perception, sensing, motion planning, and decision-making is a serious problem for reliable autonomous performance [17].

Methods for assessing the operation of neural networks and specific closed-loop safety features are provided by formal neural network verification tools such as NNV, Marabou, Verisig, and mixed-integer approaches [18]-[20]. These methods show that evaluating performance using only sample trajectories is essentially different from assessing the internal mechanisms of learned controllers or how they interact with system dynamics. The computational cost of formal analysis may increase with the model's size and the system's dynamics, even if it can produce robust guarantees under defined conditions. In order to avoid the additional complexity of neural-network verification, this research uses a symbolic controller in conjunction with a finite-state simulation environment to concentrate on the dynamics of runtime monitoring and safety shielding strategies.

2.4 Predictive Safety Certification and Navigation

Autonomous safety systems rely heavily on predictive control approaches, which incorporate anticipated system dynamics, operational constraints,

and future situations into decision-making. An approach to combining online adaptation with limited control requirements has been investigated, and it involves learning-based model predictive control [21]. An improvement to this approach is the cautious Gaussian-process model predictive control, which incorporates uncertainty into prediction and decision-making processes [22]. While set-based forecasting methods have been used for safety-critical autonomous lane-changing decisions [24], curriculum-oriented supervision has been studied as a way to reduce risky exploration in reinforcement learning [23].

Aside from constrained optimization methods and projection onto secure action sets, there are further strategies to balance task efficacy with operational sustainability [25]-[27]. Studies on safe regulation in uncertain linear-quadratic dynamics highlight the need of stability constraints and model assumptions [28]. Evidence from learning-for-exploration-based predictive control has shown how important it is to integrate prediction, adaptability, and constraint fulfillment in autonomous functioning [29]. Collectively, these investigations show that accurate environmental measurements, system dynamics, uncertainty representations, and constraint definitions are critical to predictive safety certification. Because of these considerations, the proposed RV-PS architecture combines finite-horizon risk forecasting with real-time action filtering.

2.5 Literature Synthesis and Identified Research Gap

The study establishes important foundations for formal evaluations that use simulation, runtime verification, safety shielding, control based on barrier functions, limited reinforcement learning, predictive safety evaluation, and so on. Though different in their underlying assumptions, computing needs, and evaluation contexts, existing approaches provide additional functions for limiting dangerous autonomous activities. The challenges of complex learning-capable systems and the importance of conducting thorough

safety evaluations have been highlighted by research into autonomous system frameworks and neural network validation [30], [31]. To illustrate time-extending goals and organized agent behavior, reward-machine approaches that are influenced by automata provide an alternate framework [32].

Despite these advancements, studying the interplay of connected simulation environments, runtime monitoring, uncertainty-aware prediction, action-level safety shielding, and fallback management requires a unified and fully reproducible proof-of-concept demonstration. That methodological aim is satisfied by the suggested Runtime Verification and Predictive Safety Shielding (RV-PS) approach, which uses an integrated decision pipeline to check proposed actions against safety criteria before they are executed.

Running shielding and anticipatory safety control are not claimed to be novel by the framework. Its principal contribution, within a controlled synthetic assessment scenario, is the integration of running time monitoring, uncertainty-aware finite-horizon prediction, policy-preserving action filter, and auditable fallback mechanism. Thanks to this consolidation, we can systematically examine vulnerabilities, computing needs, task performance, redundant interventions, and safety breaches in the case of environmental unpredictability. The proposed approach lays a solid foundation for the future implementation, official evaluation, and experimental verification of safety protocols for autonomous agents by expressing the assumptions and limitations of the simulation in a cohesive manner.

Table 1. Positioning of shielding and runtime verification mechanisms

Approach	Core mechanism	Strength	Boundary in current comparison
Runtime monitor	Trace check after/at event	Interpretable property state	Can detect too late
Reactive guard	Block currently observed hazards	Low conceptual complexity	Does not anticipate movement
Predictive nominal guard	Forecast and filter future positions	Anticipatory risk reduction	Depends on exact observation
Reachability guard	Exclude conservative reachable set	Strong local set-based constraint	More fallback and blocked progress
Proposed RV-PS	Observation-centered 3-step risk filter	Tunable safety-utility trade-off	Risk proxy, not certified chance bound

3. Proposed Methodology

3.1 Research design and architecture

The design follows a systematic procedure for conducting simulation studies: explain the goal of safety, establish the decision limit, evaluate candidates, create consistent scenarios, examine policy modifications, and look at counterexamples. The reference overseer, risk analyzer, and execution portal are structurally separated from the confirmed mission aim and the agent-recommended action. Aside from the execution gate, nothing changes in the virtual environment. An organized episode record contains all reactions and collision results.



Figure 1. Architectural layers of RV-PS. A heuristic policy proposes an action; a separate runtime monitor and predictive risk filter decide whether it is admissible.



Figure 2. Runtime verification workflow. Observations, specification state, action forecasting, filtering, and transition logging are evaluated at every step. 3.2 System model

A robot's position, a set of immobile obstacle cells, and three moving danger spots make up an autonomous navigation environment's discrete state s_t . The collection of actions is limited to vacant static cells inside the specified bounds and is represented by $A = \{\text{north, south, east, west, stay}\}$. The hazards move according to a seeded discrete random walk, where each mover uniformly chooses between nearby cells and staying put at each step. Every update from the robot and the movers happens at the exact same precise moment. A static distance field is used in the fundamental method to identify the nearest successor to a target, with an exploratory divergence likelihood of 0.13.

$$s_{t+1} \sim P(\cdot | s_t, a_t), \quad o_t \sim O(\cdot | s_t, \eta), \quad a_t^0 = \pi(o_t)$$

The observation o_t includes the state of each mover's reported cell. Otherwise, a report is kept accurate, and it is moved to an adjacent lawful cell with a probability $\eta \in \{0, 0.10, 0.25, 0.40\}$. An observation-centric mixed approximation is used by the shield in place of a recursively filtered posterior since the simulator is unable to learn the observation model. The comprehension of all static maps is complete. Rather than a model tailored to a particular robot or sensor, these choices indicate a purposefully intended difficulty.

3.3 Runtime safety specification and trace status

The set of occupied dynamic-risk cells immediately after the transition is denoted by U_t . Always avoiding a collision is the first and most important safety requirement; the action generator automatically removes static occupancy. While an episode is

running without additional violations, a monitor for finite-horizon traces immediately indicates a violation, stays in a pending state, and is considered satisfied once the episode ends without further collisions. This is not an argument for infinite-horizon correctness, but rather a pragmatic three-valued finite-prefix interpretation.

$$\phi_{\text{safe}} = G[0, T] (\text{robot_position}_t \notin U_t), \quad T = 42 \text{ steps.}$$

There is no sensor or motion detector built into the screen. It checks the validity of specifications using real traces, whereas the predictive shield tries to stop infractions before they happen by looking at possible actions before they happen. Accountability requires distinguishing between actual monitoring results and projected risk assessments.

3.4 Observation-centered probabilistic prediction

The prediction module divides η among the allowed non-stationary neighbors of each recognized cell and gives each detected hazard site a chance of $1 - \eta$. After that, it spreads this belief proxy using the previously-established random-walk transition kernel. Because it does not incorporate a Bayesian posterior or the whole set of observations, this belief formulation is purposefully imperfect. A three-phase nominal robot continuation is generated for each potential action using the static-map greedy technique.

$$b^0(q | o_t) = (1 - \eta)1[q = o_t] + \eta \cdot 1[q \in N(o_t)] / |N(o_t)|$$

$$b^{k+1}(q') = \sum_q b^k(q) P_{\text{mover}}(q' | q), \quad k = 0, 1, 2$$

The probability mass of each risk connected with the robot's designated cell is taken from its marginal distribution at every predicted stage. The collaborative no-collision proxy takes into account the simulated movers' marginal non-collision probabilities across different time intervals. The product provides an approximation, not a hard cap, on the actual possibility of a collision because of the consistency of predictions across time.

$$\hat{r}(a | o_t) = 1 - \prod_{k=1}^H [1 - b_{m,k}(\text{robot_cell}^k(a))], \quad H = 3$$

3.5 Shield policy and fallback

The recommended shield assesses a risk substitute for each legally sanctioned robotic operation. As long as the proxy risk associated with an action stays below $\tau=0.075$, it is considered legal. We keep the underlying proposition when it's acceptable. As an alternative, the shield chooses a cost-effective measure from the available possibilities based on the vicinity of the fixed target and the estimated risk. When the threshold is not exceeded, the shield falls back to a minimum-modeled-risk mode; however, this mode does not have the necessary independent certification to prevent collisions. Improved career chances come with a trade-off: a less stringent barrier might sanction riskier pursuits. Instead of being independently determined safety standards, all experimental thresholds are designed parameters of the simulation.

$$A\tau(o_t) = \{a \in A_{\text{legal}}(s_t) : \hat{r}(a | o_t) \leq \tau\}$$

$$a_t = a_t^0 \text{ if } a_t^0 \in A\tau; \text{ otherwise } \operatorname{argmin}_{a \in A\tau} [d_{\text{goal}}(a) + 2\hat{r}(a)]$$

The implementation selects a lawful action that minimizes $\hat{r}(a) + 0.02d_{\text{goal}}(a)$ while $A\tau$ is empty. The selection method is defined because the candidate set always includes a stay action; nevertheless, when a moving obstacle enters the robot cell, the stay action itself can be hazardous. Whenever there is a change from the first action or a period when the acceptable set is empty, the code notes it.



Figure 3. Diagram for making simplified permission decisions.

3.6 Conditional safety property

A conventional one-step safety argument can be formulated under stricter conditions than those utilized by the operational probability filter. Envision a scenario where the monitor receives a sound set R_t including all potential future danger positions, evaluates the planned robot successor in relation to R_t , and always has a separate validated backup successor outside of R_t . Starting from a safe state, avoiding collisions beyond the specified horizon is maintained by induction by removing all possible successors in R_t . This assertion is conditional on the following: the availability of a safe contingency, the accuracy of state localization, the dependability of action execution, and the completeness of the model.

3.7 Pseudocode and complexity

Algorithm 1. RV-PS action mediation environment:

Input: static map, robot position, observed hazard cells, proposal a_0 , η , horizon H , threshold τ .

1. Compute all legal successor actions and their distance-to-goal estimates.
2. For each hazard, build the observation-centered distribution and propagate H steps.
3. For each candidate, roll out the greedy continuation and compute marginal-risk proxy $\hat{r}$.
4. Keep actions whose estimated risk does not exceed τ .
5. If a_0 passes, execute a_0 ; else choose the lowest-cost passing alternative.
6. If no action passes, choose a least-modeled-risk fallback; record the fallback event.
7. Advance the common simulated world; update the monitor verdict and log outcome.

The fundamental sparse propagation, in a grid with $|S|$ cells, B possible robot actions, M risks, and horizon H , always looks at local outgoing transitions of the possible hazard belief supports. Without considering map-distance preprocessing, a rough upper limit is $O(BMH|S|\Delta)$, where $\Delta \leq 5$ is the random-walk branching factor. This is not a measured hardware latency, but rather an algorithmic maximum. Issues with execution time, memory allocation, and meeting deadlines are still present in the deployment process.

3.8 Evaluation metrics

The robot is considered to have collided whenever it shared a post-transition cell with another mover. How many episodes reach the goal without a collision within 42 acts is called task completion. When protected action departs from the core proposal, intervention count documents those instances. When the policy-specific conditions have not been satisfied, the fallback count is logged. While proximal events record transition-related motions within one Manhattan block of a mover, average step count factors in overall path length, delays, and collision reductions. Because a too-tight barrier could impede navigation and prevent collisions, it is important to examine the frequency of collisions alongside completion.

$$\text{CollisionRate} = (\text{number of collision episodes})/(\text{number of episodes})$$

$$\text{CompletionRate} = (\text{collision-free goal-reaching episodes})/(\text{number of episodes})$$

4. Experimental Design and Simulation Setup

4.1 Study type and central disclosure

We evaluate the proposed framework's viability by deriving the experimental results from the simulation parameters.

This study makes use of an authentic seeded Python emulator. Only within the specified virtual worlds are quantitative rates measured. The results can be seen as estimates of the collision frequencies in real implementations, since the experimental setup and environmental distribution were created for the purpose of evaluating mechanisms.



Figure 4. Study pipeline. Each generated world is reused across all policy variants, and random hazard trajectories are held fixed for paired comparison.

4.2 Environment and dataset

In each map, there is a 12×12 grid with 15 randomly selected fixed obstacles. If the normal configuration of random obstacles separates the starting point from the ending point, the program will remove those obstacles for that scenario. The computer starts its ascent from the top left cell and aims to reach the bottom right cell. Legal random walks are performed by three autonomously traversing barriers. When the objective is reached, a collision happens, or 42 actions have occurred, the run terminates. There are 960 possible permutations of world and noise, with 240 worlds generated independently for each sensor noise parameter. For each domain, the identical hidden paths and data are used across all five methods, leading to 4,800 iterations of paired policies.

Table 2. Explicit simulator configuration.

Parameter	Setting	Interpretation
Map and horizon	12×12 ; 42 steps	Finite discrete navigation
Moving hazards	3 random walkers	Shared exogenous disturbance sequences
Static walls	15 candidates	Exact obstacle knowledge

Parameter	Setting	Interpretation
Worlds per noise	240	960 paired worlds across four noise conditions
Observation error η	0, 0.10, 0.25, 0.40	Reported hazard shifted to adjacent legal cell
Exploration rate	0.13	Base greedy policy occasionally deviates
Predictive ahead	look- H=3	Current proposal plus 2 greedy continuation steps
Proposed threshold	risk $\tau=0.075$	Heuristic marginal-risk cutoff
Simulation seed	20261013	Deterministic episode construction

4.3 Baselines and fair comparison

In this study, we compare five different policies using a single base controller and constant global observations: B0 is the basic proposal without runtime protection, B1 is a responsive runtime safeguard that ignores currently identified hazard cells, B2 is a standard three-step safeguard that prevents actions whose greedy continuation encroaches upon a currently noted hazard cell, B3 is a cautious reachability-oriented safeguard that disallows subsequent actions within a Manhattan distance of two from any reported hazard, and RV-PS is the proposed observation-centric probabilistic three-step shield. Differences in outcomes are not due to varied learned tactics but rather to action mediation, since all methods use the same static map and allowed robot motions.

Unlike the complex published shield synthesizers, barrier-certified RL algorithms, and formal verification tools discussed in Section 2, these comparators purposefully simplify mechanisms. The reachability guard is not a fully-state winning-set controller that is generated automatically; it is only a geometric overapproximation for demonstration

purposes.

4.4 Sensitivity protocol

Maintaining the policy specifications in tact, the primary sensitivity experiment modifies the noise of the observations across all four settings. At $\eta=0.25$, the same 240 worlds are kept and each RV-PS component is changed in turn: disregard observational uncertainty in predictions; restrict look-ahead to $H=1$; change τ to 0.20; shrink τ to 0.025; or replace minimum-risk fallback with a wait-only fallback.

4.5 Reproducibility strategy

The provided reproducibility package includes a Python/NumPy simulator, raw CSV data at the episode level, an independently generated ablation CSV, a JSON summary, and scripts for visualization. With the exception of the shield, the identification of the incident is necessary for the universal seed to function, leading to interdependent pathways for external threats across policies. In a given noise environment, the data used for comparisons and the observation seed are identical.

5. Results and Analysis

5.1 Comparison across all conditions

Table 3. Executed results pooled over 960 paired worlds per method.

Method	Collision % ↓	Completion % ↑	Interventions	Fallback events	Mean steps
B0: unshielded	24.17	75.83	0.00	0.00	20.98
B1: reactive RV	25.31	74.69	0.38	0.00	21.19
B2: nominal H=3	16.98	82.50	1.34	0.16	23.28
B3: reachability	9.90	70.83	7.46	1.10	30.26
RV-PS: proposed	2.08	71.35	7.63	1.47	31.52

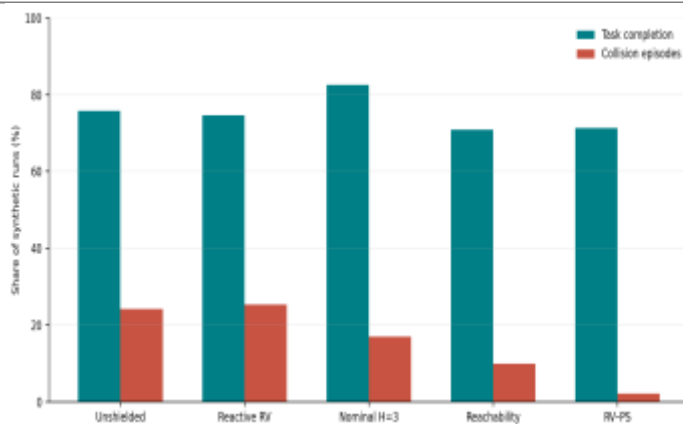


Figure 5. Performance comparison. The proposed shield has the lowest collision rate but does not maximize completion, making the safety–utility trade-off explicit.

Among the occurrences where an unshielded control reaches the target, 24.17% have a collision with it. While the reactive rule does not improve the collision rate by 25.31 percent, action replacements can direct the robot to an occupied cell later on when only the present observations are taken into account. Crash rates drop to 16.98% and completion rates rise to 82.50% with nominal three-step exclusion. The geometric reachability guard achieves just 70.83 percent of tasks, but it successfully decreases collisions to 9.90 percent. The average accident rate is reduced to 2.08% by RV-PS, with a completion rate of 71.35% and 7.63 interventions per episode. Results show that expected risk distributions can direct pre-execution filtering, which is in line with the suggested method, and that there are substantial functional costs associated with the strict 0.075 criterion.

5.2 Sensitivity to imperfect observations

Table 4. Simulation sensitivity to inaccurate mover localization. Fixed worlds and hazard paths are reused across methods at each noise level.

Sensor noise %	B0 collision %	B2 collision %	B3 collision %	RV-PS collision %	RV-PS completion %
0	24.17	16.67	3.33	0.83	73.33
10	24.17	17.50	5.42	2.08	70.42
25	24.17	17.08	15.00	2.92	73.75
40	24.17	16.67	15.83	2.50	67.92

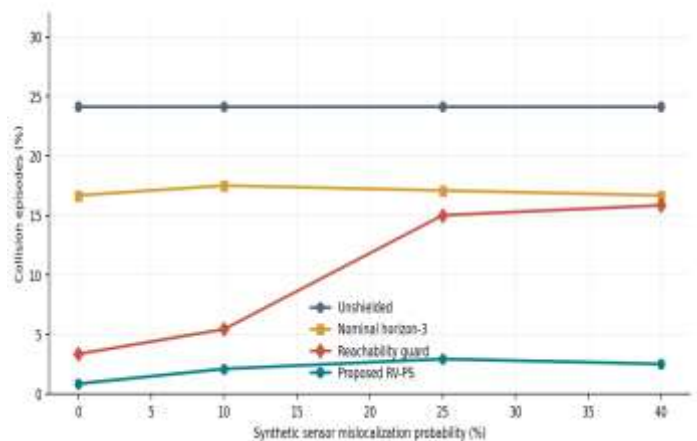


Figure 6. Collision sensitivity with observation error. Reachability-style exclusion becomes less effective when reported hazard positions are displaced, while the proposed predictor explicitly redistributes observation probability.

At an error level of 40%, RV-PS reports 2.50% of collision events, compared to 0.83% at a zero observation error. If the reported location is not in line with the actual obstacle, the selected substitute or backup action could still offer dangers, so the reachability guard increases from 3.33% to 15.83%. There is a non-monotonic trend in the RV-PS completion: 73.75% at 25% noise and 67.92% at 40% noise. It would be unwise to assume a monotone empirical curve formed from four minimal design points when dealing with stochastic routes for prediction and action selection. The main policy in this experiment uses exact static-map distance instead of risk observations, therefore noise has no effect on the unshielded rate.

5.3 Component ablation and threshold study

Table 5. Paired ablations at $\eta=0.25$, 240 common worlds. Percentage-point changes arise from changed rule parameters.

RV-PS configuration	Collision %	Completion %	Interventions	Fallbacks
Full RV-PS	2.92	73.75	7.28	1.57
No observation uncertainty	3.33	75.83	6.85	1.39
One-step risk (H=1)	11.25	87.92	1.98	0.11
Loose threshold (0.20)	4.17	88.75	3.66	0.60
Conservative threshold (0.025)	0.42	56.67	11.45	3.44
Wait-only fallback	11.67	68.33	8.07	2.68

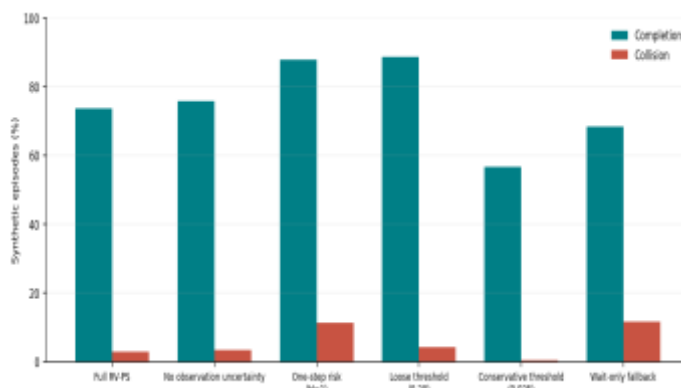


Figure 7. Ablation results. A conservative cutoff minimizes modeled collisions but reduces task completion; a looser threshold restores progress with higher risk.

The collision frequency increases from 2.92% to 11.25% and the completion rate goes up from 73.75% to 87.92% when the look-ahead is reduced to one step. This establishes horizon depth as an essential component in improving operational strain and safety. Crash rates increase to 3.33% and completion rates to 75.83% when the assumed observation uncertainty is set to zero. With a risk level of 0.20, completion is improved to 88.75% but collisions are increased to 4.17%; with a risk threshold of 0.025, collisions are reduced to 0.42% but completion is

reduced to 56.67%. When dynamic items can invade a stationary robot's zone, the wait-only fallback increases collision rates to 11.67%, showing that a seemingly careful stopping criterion may not be secure. Please note that these implications are exclusive to this transition model and do not reflect the treatment effects that are expected for general autonomous agents.

5.4 Execution burden and workflow efficiency

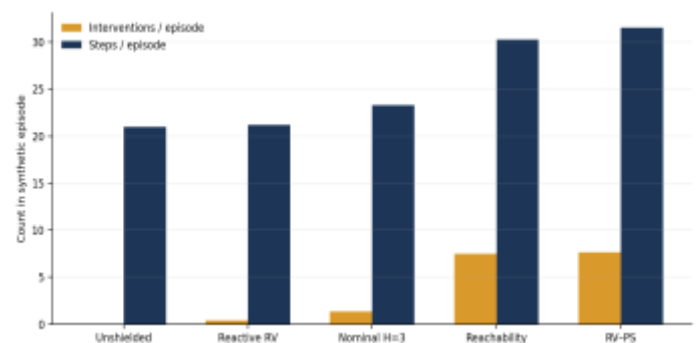


Figure 8. Workflow-efficiency proxies. Bars represent actual simulator intervention and episode-step counts.

The recommended shield has an average of approximately 31.52 decisions, in contrast to the nearly 21 decisions made by the unconstrained and reactive strategies. The nominal forecast averages around 1.34 episodes of intervention load and the cautious reachability guard averages about 7.46 episodes of intervention burden. Despite the rarity of crashes, these supplementary options show how satisfaction could decrease. Real simulated action sequences are represented by the counts, which are not easily convertible to milliseconds, energy consumption, or network overhead. Section 3.7 explains that the basic prediction cost increases as the number of supported grid cells, risks, possible actions, and time horizon increases.

5.5 Error and residual-risk analysis

Table 6. Structured interpretation of residual failures and omissions in the constructed simulation.

Observed limitation	Mechanism in simulated system	Implication
Residual RV-PS collisions	Approximate marginal risk and a non-certified fallback	No absolute safety guarantee
Low completion at strict τ	Future occupancy proxy excludes productive moves	Utility must accompany collision rate
Wait-only fallback failures	Hazards can move into a stationary robot	Stopping is not automatically safe
Noise mismatch	Observed hazard can be displaced from true position	Model coverage is conditional
Nominal guard blind spots	Static hazard assumption omits motion	Prediction model choice matters
Path cost in safe policies	Detours and waiting consume step budget	Intervention cost matters

The main error is the difference between the officially sanctioned limit and the assessed collision risk. Even though no action satisfies the criterion, the alternative that was proposed nevertheless executes an action. In addition, the resulting estimate does not precisely guaranty a maximum limit on the actual likelihood of occurrence because the product of incremental marginal non-collision probabilities does not take into account the temporal interdependence within the trajectory of each individual hazard. Instead of calling the predictor "provably safe," the study shows the actual collision frequencies. An observation model that has been confirmed, an analysis of all reachable sets, and a certified invariant set backup plan are all required for a prospective system.

5.6 Interpretation relative to research questions

RQ1 is met with a conditional affirmative reply: in relation to the identified comparators, three-step risk-aware shielding markedly lessens collision incidents.

RQ2 presents a nuanced outcome: more rigorous screening enhances intervention and step totals but constrains target achievement, whereas lenient criteria reestablish functionality at a heightened risk of collisions.

RQ3 is corroborated by sensitivity and ablation data indicating that the management of uncertainty, duration of the horizon, risk appetite, and fallback options are all significant. These assertions pertain to the simulated decision-making framework, rather than to external deep reinforcement learning standards or actual robotic implementations.

6. Discussion

Buildings are the principal target of the effect. A safeguarding mechanism sets a protective execution limit, while runtime verification evaluates any violations of operational safety properties in a trace-oriented manner. Their isolation clarifies the proof. The shield could provide the impression that a collision is less likely than it actually is, even though a trace monitor would have correctly identified it. The generated partition improves auditability: without determining the model's intrinsic purpose, an auditor may reproduce the proposed action, expected distribution of hazards, allowable range of alternatives, selected replacement, and actual transition between states.

The simulation also shows that there might be different trade-offs with nominal and uncertainty-aware predictors. Improving model conservatism alone won't make missions more effective, and finding the controller with the lowest collision frequency isn't necessarily the best option. No robot can complete its tasks in the allotted time if it hesitates or chooses a risk-averse but mission-diverting course of action on a regular basis. Therefore, before choosing a shield threshold, a

production design needs to define explicit parameters for fault management, acceptable intervention costs, deadline importance, and risk tolerance.

This framework suggests a sequential engineering approach for practitioners to follow: identify critical physical dangers, set safety assertions, outline the observation and transition model, determine the available secure successors, assess the accuracy of risk assessments, deploy an independently verified contingency, and document every successful action. Runtime verification, predictive certification, learning control, and scenario evaluation are all covered in the literature from 2019 to 2020 [1–5], [15–16], [21–25]. The proposed proof of concept does not claim a revolutionary formal-synthesis method, but rather combines these ideas into a simple executable decision loop.

For methodological reasons, the deterministic paired world design is preferable, since different shields face the same external risk trajectories. Despite this, controllers can still change their routes and encounter new threats while using trajectory pairing to reduce model misspecification. In the specified gridworld, a conditional action is the obvious result. Before reaching any conclusions at the deployment level, it is necessary to independently evaluate the variety of situations, the correctness of formal abstractions, the varieties of perception errors, and the deadlines for execution.

7. Limitations and Future Work

The simulator includes three dynamic dangers that are modeled independently and one navigation agent. It doesn't take into account things like uncertainty in learned policies, occlusion, multi-robot collaboration, vehicle motion, unbroken trajectories, actuator latency, or collision dynamics among physical entities. Although the agent controls the perils, the real players can behave strategically, be completely immobile, or respond to the agent's every move. Real safety-critical procedures cannot be represented with

a 12×12 grid and a 42-step episode constraint.

The simulated sensor outputs are generated by the same scalar observation-error parameter, which the belief proxy assumes to be the same. It doesn't check if that parameter is constant or deducible. There is no documented safety in the backup plan, and the forward risk instrument doesn't take time correlation into account. Therefore, the zero-collision guaranty of a theoretically sound reachable-set shield and the 2.08% estimated collision rate are not safety guaranties for the real system. Additionally, the present investigation does not evaluate scheduler latency, processor overhead, hardware energy usage, network transmission, or the feasibility of running the test at real control frequencies.

Research in the future should start with using a real posterior estimator and careful probability bounds, then build an action shield from a formal safety game or a strong reachable-set design with a proven backup. The quantity of barriers, velocity distributions, occlusion configurations, uncertainty model inconsistencies, and decision-making time limits should all be adjusted in future tests. To determine if the method can withstand the effects of implementation, controlled physical laboratory trials with emergency-stop measures would follow, followed by a representative robotics simulator such as ROS/Gazebo. Rather of overseeing robots independently, multi-agent systems need handle interdependent behaviors, deadlock, and fairness constraints.

Finally, the next quantitative evaluations should compare to real shielding and barrier-based benchmarks in the same models and settings, show confidence intervals in different contexts, and evaluate the full utility-risk Pareto frontier rather than just one preferred threshold. It would be less likely to modify the benchmark to fit the proposed policy if the methodology and autonomous testing suite were pre-registered.

8. Conclusion

This research has revealed RV-PS, a design-science framework that integrates runtime temporal monitoring, observation-focused finite-horizon risk assessment, action filtration, and backup mediation for autonomous navigation systems. Five policies were examined in an authentically executed gridworld review, which yielded 4,800 paired policy trials, across 960 typical world/noise scenarios. According to its stated assumptions, RV-PS reduced the aggregate collision-episode rate from 24.17% without shielding to 2.08%, while simultaneously lowering completion to 71.35%. Reducing the prediction horizon or risk threshold speeds up progress but increases the likelihood of modeling safety violations; choosing an uncertified wait-only fallback could inevitably lead to crashes, according to studies. These findings are not indicative of operational safety or real-world excellence; rather, they are replicable outcomes from a designated simulator. Included in the contribution are a set of well-defined safety-utility assumptions, a realistic evaluation baseline, and a transparent methodology. It is necessary to convert the preliminary risk assessment into verified uncertainty parameters and verified backup plans before studying the resultant system in more complex simulations and hardware-in-the-loop environments.

REFERENCES

- [1] C. Sánchez et al., “A survey of challenges for runtime verification from advanced application domains (beyond software),” *Formal Methods in System Design*, vol. 54, pp. 279–335, 2019. <https://doi.org/10.1007/s10703-019-00337-w>
- [2] G. Avni, R. Bloem, K. Chatterjee, T. A. Henzinger, B. Könighofer, and S. Pranger, “Run-Time Optimization for Learned Controllers Through Quantitative Games,” in *Computer Aided Verification (CAV)*, LNCS 11561, pp. 630–649, 2019. https://doi.org/10.1007/978-3-030-25540-4_36
- [3] S. Bharadwaj, R. Bloem, R. Dimitrova, B. Könighofer, and U. Topcu, “Synthesis of Minimum-Cost Shields for Multi-agent Systems,” in *American Control Conference*, pp. 1048–1055, 2019. <https://doi.org/10.23919/ACC.2019.8815233>
- [4] N. Jansen, B. Könighofer, S. Junges, A. Serban, and R. Bloem, “Safe Reinforcement Learning Using Probabilistic Shields,” in *CONCUR, LIPIcs* 171, pp. 3:1–3:16, 2020. <https://doi.org/10.4230/LIPIcs.CONCUR.2020.3>
- [5] B. Könighofer, F. Lorber, N. Jansen, and R. Bloem, “Shield Synthesis for Reinforcement Learning,” in *ISoLA, LNCS* 12476, pp. 290–306, 2020. https://doi.org/10.1007/978-3-030-61362-4_16
- [6] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, “End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks,” in *AAAI*, vol. 33, pp. 3387–3395, 2019. <https://doi.org/10.1609/aaai.v33i01.33013387>
- [7] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, “A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems,” *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2019. <https://doi.org/10.1109/TAC.2018.2876389>
- [8] J. Choi, F. Castañeda, C. J. Tomlin, and K. Sreenath, “Reinforcement Learning for Safety-Critical Control under Model Uncertainty, using Control Lyapunov Functions and Control Barrier Functions,” in *Robotics: Science and Systems*, 2020. <https://roboticsconference.org/2020/program/papers/88.html>
- [9] A. Taylor, A. Singletary, Y. Yue, and A. Ames, “Learning for Safety-Critical Control with Control Barrier Functions,” in *Learning for Dynamics and Control*, PMLR 120, pp. 708–717, 2020. <https://proceedings.mlr.press/v120/taylor20a.html>
- [10] A. Stooke, J. Achiam, and P. Abbeel, “Responsive Safety in Reinforcement Learning by PID Lagrangian Methods,” in *ICML, PMLR* 119, pp. 9133–9143, 2020. <https://proceedings.mlr.press/v119/stooke20a.html>
- [11] A. Wachi and Y. Sui, “Safe Reinforcement Learning in Constrained Markov Decision Processes,” in *ICML, PMLR* 119, pp. 9797–9806, 2020.

<https://proceedings.mlr.press/v119/wachi20a.html>

[12] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge, "Projection-Based Constrained Policy Optimization," in ICLR, 2020. <https://mlanthology.org/iclr/2020/yang2020iclr-projectionbased/>

[13] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward Constrained Policy Optimization," in ICLR, 2019.

<https://mlanthology.org/iclr/2019/tessler2019iclr-reward/>

[14] A. Ray, J. Achiam, and D. Amodei, "Benchmarking Safe Exploration in Deep Reinforcement Learning," arXiv preprint, 2019. <https://openai.com/index/benchmarking-safe-exploration-in-deep-reinforcement-learning/>

[15] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. A. Seshia, "VerifAI: A Toolkit for the Formal Design and Analysis of Artificial Intelligence-Based Systems," in CAV, LNCS 11561, pp. 432–442, 2019. https://doi.org/10.1007/978-3-030-25540-4_25

[16] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on Scenario-Based Safety Assessment of Automated Vehicles," IEEE Access, vol. 8, pp. 87456–87477, 2020. <https://doi.org/10.1109/ACCESS.2020.2993730>

[17] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," IEEE Access, vol. 8, pp. 58443–58469, 2020. <https://doi.org/10.1109/ACCESS.2020.2983149>

[18] H.-D. Tran, X. Yang, D. M. Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson, "NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems," in CAV, LNCS 12224, pp. 3–17, 2020. https://doi.org/10.1007/978-3-030-53288-8_1

[19] G. Katz et al., "The Marabou Framework for Verification and Analysis of Deep Neural Networks," in CAV, LNCS 11561, pp. 443–452, 2019. https://doi.org/10.1007/978-3-030-25540-4_26

[20] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and

I. Lee, "Verisig: Verifying Safety Properties of Hybrid Systems with Neural Network Controllers," in HSCC, pp. 169–178, 2019.

<https://doi.org/10.1145/3302504.3311806>

[21] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-Based Model Predictive Control: Toward Safe Learning in Control," Annual Review of Control, Robotics, and Autonomous Systems, vol. 3, pp. 269–296, 2020. <https://doi.org/10.1146/annurev-control-090419-075625>

[22] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious Model Predictive Control Using Gaussian Process Regression," IEEE Transactions on Control Systems Technology, vol. 28, no. 6, pp. 2736–2743, 2020. <https://doi.org/10.1109/TCST.2019.2949757>

[23] M. Turchetta, A. Kolobov, S. Shah, A. Krause, and A. Agarwal, "Safe Reinforcement Learning via Curriculum Induction," in Advances in Neural Information Processing Systems, vol. 33, 2020. https://papers.neurips.cc/paper_files/paper/2020/hash/8df6a65941e4c9da40a4fb899de65c55-Abstract.html

[24] H. Krasowski, X. Wang, and M. Althoff, "Safe Reinforcement Learning for Autonomous Lane Changing Using Set-Based Prediction," in IEEE Intelligent Transportation Systems Conference (ITSC), 2020.

<https://doi.org/10.1109/ITSC45102.2020.9294259>

[25] S. Gros, M. Zanon, and A. Bemporad, "Safe Reinforcement Learning via Projection on a Safe Set: How to Achieve Optimality?" IFAC-PapersOnLine, vol. 53, no. 2, pp. 8076–8081, 2020. <https://doi.org/10.1016/j.ifacol.2020.12.2276>

[26] Y. Liu, J. Ding, and X. Liu, "IPO: Interior-Point Policy Optimization under Constraints," in AAAI, vol. 34, no. 4, pp. 4940–4947, 2020. <https://doi.org/10.1609/aaai.v34i04.5932>

[27] Y. Zhang, Q. Vuong, and K. W. Ross, "First Order Constrained Optimization in Policy Space," in Advances in Neural Information Processing Systems, vol. 33, 2020. <https://proceedings.neurips.cc/paper/2020/file/af5d5ef24881f3c3049a7b9bfe74d58b-Paper.pdf>

- [28] S. Dean, S. Tu, N. Matni, and B. Recht, "Safely Learning to Control the Constrained Linear Quadratic Regulator," in American Control Conference, pp. 5582–5588, 2019. <https://doi.org/10.23919/ACC.2019.8814865>
- [29] T. Koller, F. Berkenkamp, M. Turchetta, J. Boedecker, and A. Krause, "Learning-based Model Predictive Control for Safe Exploration and Reinforcement Learning," arXiv preprint arXiv:1906.12189, 2019. <https://arxiv.org/abs/1906.12189>
- [30] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A Survey of Deep Learning Techniques for Autonomous Driving," Journal of Field Robotics, vol. 37, no. 3, pp. 362–386, 2020. <https://doi.org/10.1002/rob.21918>
- [31] V. Tjeng, K. Y. Xiao, and R. Tedrake, "Evaluating Robustness of Neural Networks with Mixed Integer Programming," in ICLR, 2019. <https://dblp.org/rec/conf/iclr/TjengXT19.html>
- [32] R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning," arXiv preprint arXiv:2010.03950, 2020. <https://arxiv.org/abs/2010.03950>