Deductive Verification vs. Model Checking for Physical Safety of a Feedback-Controlled Drone

Can Afacan Tianrun (Eric) Yang

Abstract

We investigate the tradeoffs between different methods in formal verification for proving the safety and liveness properties of a closed-loop controlled physical systems. In particular, we will mathematically model and study a drone moving in a 3D space controlled by a hardware-oriented controller that keeps it within a safe region, and analyze the resulting closed-loop system using three approaches: explicit-state model checking in TLA+/TLC, symbolic model checking in nuSMV, and deductive proof in Lean 4. We also study the model across two tiers of complexity, with each tier introducing richer physics, more involved controller behavior, etc. This allows us to compare the methods in terms of specification expressiveness, guarantee strength, proof effort, and scalability as the system becomes more complex.

1 Introduction

Controller-driven closed-loop physical systems are ubiquitous in the real world, from industrial robot arms to flight control on commercial airplanes. These systems are more broadly instances of cyber-physical systems (CPS) (Rajkumar et al., 2010), in which a discrete computing and communication core controls and coordinates its operations.

CPS are highly safety critical, due to the nature of their direct interactions with humans and their physical environment. With the emergence of interactive robots that continuously faces the challenge of ensuring safety when deployed into real life environments (Fauna Robotics et al., 2026) , the task of ensuring that controlled physical systems either via traditional feedback controllers or Reinforcement Learning training follow safety rules is more important than ever.

To quote Dijkstra,"program testing can be used to show the presence of bugs, but never to show their absence" (Dijkstra, 1970). Unit testing and exhaustive tests can only ever expose the existence of errors, which is not good enough in our case.

Formal verification and specification offer powerful tools and methods to achieve safety verification through specifying a system in terms of mathematics. In particular, safety properties are encoded in Linear Temporal Logic (LTL), and a specific system can be checked against a specification using deduction rules in formal logic to provide guarantees.

Two paradigms particularly stand out: deductive verification and model checking. The former has historically required substantial effort and time from experts educated in logic (Clarke et al., 1999). This is now changing with the rise of autoformalization and theorem-proving assistant tools in the language of Lean, such as Gauss, which formalized parts of Maryna Viazovska's famous sphere-packing problem in dimension 8 that yielded a Fields Medal (Hariharan et al., 2026). There are also tactic finders such as LeanDojo (Yang et al., 2023) and Canonical (Norman and Avigad, 2025). The future landscape of deductive verification looks promising.

Model checking is an automatic verification technique for finite-state systems: a model checker explores the reachable behaviors of a transition system and checks whether they satisfy a temporal-logic specification (Clarke et al., 1986). Model checkers always terminate with a definitive yes/no

answer regarding whether a system meets its spec, given enough resources. (Clarke et al., 1999) In explicit-state model checking, reachable states are represented and explored directly, which often makes counterexample traces easy to interpret but can suffer from state explosion; tools such as SPIN illustrate this explicit/on-the-fly style of verification (Holzmann, 1997). In symbolic model checking, large sets of states and transitions are represented compactly using symbolic structures such as Boolean formulas or binary decision diagrams, allowing some systems with very large state spaces to be analyzed more efficiently (Burch et al., 1992). This distinction is central to our comparison between TLA+/TLC and nuSMV: TLC gives a direct explicit-state exploration of the fixed-point drone model, while nuSMV provides a symbolic encoding that may scale differently as the state space grows.

In the existing literature for formal verification, there are case studies that compare different model checkers (Araiza-Illan and Eder, 2015; Mazzanti and Ferrari, 2018), but few papers that make direct comparison between deductive verification and model checking specifically on the task of verifying physical safety.

This paper aims to fill that gap by directly comparing model checking against deductive proof using the Lean 4 theorem prover, verifying LTL safety properties on a controller-based drone model. We will model the physics and behavior of the drone mathematically, using a hardware-oriented controller. Feedback-controlled drones are examples of cyber-physical systems, where software or hardware controllers interact with physical dynamics. Furthermore, they are considered to be hybrid systems, which are systems that involve the interaction of discrete and continuous dynamics (Lygeros et al., 1999). The actual dynamics of a real life drone would be continuous, yet they are controlled by discrete controllers, and in particular in this paper, their physical laws will follow discrete update rules.

1.1 Contributions

- We define a unified sampled-data drone model with a finite-valued guard-band controller and fixed-point state representation.
- We use the principles outlined in Lygeros et al. (1999) to design verifiable controllers that are the least restrictive and hardware-oriented.
- We verify the same baseline properties: geofence safety, speed boundedness, and recovery from the guard region, using explicit-state model checking in TLA+/TLC, symbolic model checking in nuSMV, and deductive proof in Lean 4.
- We introduce a nonconvex safe region and augmented controller logic for Tier 2, verify the finite fixed-point Tier 2 model in TLA+/TLC and nuSMV, and formalize corresponding Tier 2 safety obligations in Lean 4.
- We implement synthesizable SystemVerilog versions of the Tier 1 and Tier 2 fixed-point closed-loop systems. This RTL implementation serves as a hardware-oriented counterpart to the TLA+, nuSMV, and Lean models. The controller is implemented as finite-valued threshold logic, while the dynamics module implements the fixed-point sampled-data update using integer arithmetic. This allows us to test whether the mathematical transition system used for verification can be mapped to realistic RTL-style logic. In addition, we include a SystemVerilog safety monitor and testbench. The monitor checks the fixed-point versions of the three baseline properties: geofence safety, speed boundedness, and recovery from the guard region. Since simulation is finite, the unbounded LTL recovery property is approximated as

bounded recovery within a configurable number of cycles. Directed tests check hover behavior, controller threshold behavior, and recovery from the guard band, while randomized tests explore many admissible initial states on the fixed-point lattice.

- We compare the approaches in terms of expressiveness, guarantee strength, proof/debugging effort, scalability, and hardware relevance.

2 Drone Model

Our drone model is a point mass in $3D$ space equipped with a geofence region and a guard region that the drone is permitted to move within. The controller is a deterministic finite-valued guard-band controller that applies inward corrective thrust when the drone is near the guard region, and otherwise applies a normalized hover thrust to keep the drone floating.

The full mathematical derivations of the continuous dynamics, sampled-data update, controller, and temporal-logic properties summarized in this section are given in Section A.

2.1 Closed-Loop Model

We verify the sampled closed-loop system

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) = F(\mathbf{x}_k, g(\mathbf{x}_k)), \qquad \mathbf{x}_k = (\mathbf{q}_k, \mathbf{v}_k) \in \mathbb{R}^3 \times \mathbb{R}^3. \tag{1}$$

where $\mathbf{q}_k$ and $\mathbf{v}_k$ denote the position and velocity of the drone, respectively.

The sampled-data update is

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \Delta t\, \mathbf{v}_k + \frac{1}{2}\Delta t^2(\mathbf{u}_k - \mathbf{e}_3), \tag{2}$$

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \Delta t(\mathbf{u}_k - \mathbf{e}_3). \tag{3}$$

where $g : X \to U$ is the deterministic guard-band controller. Writing the geofenced safe region as the axis-aligned box S, the inner safe set as S_{in}, and the guard region as $B = S \setminus S_{\text{in}}$ with guard-band widths $\delta_h, \delta_3^-, \delta_3^+ > 0$, the control $g(\mathbf{q}, \mathbf{v}) = (u_1, u_2, u_3)$ applies inward corrective thrust on each axis whose position lies in B with outward velocity, and the normalized hover input otherwise:

$$u_i = \begin{cases} +u_h & q_i \le q_i^{\min} + \delta_h,\ v_i < 0, \\ -u_h & q_i \ge q_i^{\max} - \delta_h,\ v_i > 0, \\ 0 & \text{otherwise}, \end{cases} \qquad i = 1, 2,$$

$$u_3 = \begin{cases} u_3^{\max} & q_3 \le q_3^{\min} + \delta_3^-,\ v_3 < 0, \\ u_3^{\min} & q_3 \ge q_3^{\max} - \delta_3^+,\ v_3 > 0, \\ u_{\text{hov}} & \text{otherwise}. \end{cases}$$

The outputs are drawn from a finite control alphabet $U_{\text{ctrl}} \subseteq U$. The full construction, finite alphabet, and well-definedness of g are given in Section A.3.

2.2 Safety and Liveness Properties

The primary safety and liveness properties we verify are

$$\square(\mathbf{q} \in S), \tag{4}$$

$$\square(\|\mathbf{v}\| \leq v_{\max}), \tag{5}$$

$$\square(\mathbf{q} \in B \rightarrow \Diamond(\mathbf{q} \in S_{\text{in}})). \tag{6}$$

Here S, S_{in}, and B are the geofenced safe region, inner safe set, and guard region of Section 2.1, and $v_{\max}$ is the speed bound. They specify geofence safety, boundedness of speed, and eventual recovery from the guard region, respectively.

2.3 Complexity Tiers

We study two levels of system complexity. Each tier retains the baseline closed-loop structure in (1), the sampled update rules in (2)–(3), and the baseline properties in (4)–(6), while adding new dynamics, controller logic, or safety/liveness properties.

2.3.1 Tier 1: Baseline

The Tier 1 system is the baseline point-mass drone with gravity, with the guard-band controller g. The safe region S is an axis-aligned box, the controller uses a finite control alphabet U_{ctrl}, and the properties to verify are geofence safety, speed boundedness, and recovery from the guard region.

For the finite-state model-checking analysis, we fix the fixed-point scale and sampling period, and compare the Tier 1 and Tier 2 reachable state spaces, verification runtimes, and symbolic diagnostics. In contrast, the Lean formalization states the sampled-data obligations parametrically, with timestep conditions appearing as explicit hypotheses where required.

2.3.2 Tier 2: Obstacles

Tier 2 extends the baseline geofence model by adding an axis-aligned open no-fly box

$$O = (o_1^{\min}, o_1^{\max}) \times (o_2^{\min}, o_2^{\max}) \times (o_3^{\min}, o_3^{\max}) \subset S_{\text{in}}.$$

The safe position region is therefore $S \setminus O$. Since O is open, points on ∂O are treated as safe.

The sampled dynamics F and finite control alphabet are unchanged. We use an augmented controller g_2, which adds obstacle-avoidance guard bands around the faces of O. These guard bands apply corrective thrust away from the obstacle when the drone is aligned with O and moving toward it. The Tier 2 closed-loop system is

$$f_2(x) = F(x, g_2(x)).$$

In addition to the Tier 1 properties, we check the obstacle-avoidance invariant

$$\textbf{P4.} \qquad \square\,(\mathbf{q} \notin O).$$

The detailed obstacle geometry, guard-band conditions, and fixed-point encoding are given in Section C.1.

2.4 SystemVerilog RTL Implementation and Simulation Setup

We also implement the fixed-point closed-loop models in synthesizable SystemVerilog. The RTL is not treated as a fourth formal-verification method. Instead, it serves as an implementation-correspondence layer: it checks whether the transition systems specified in TLA+/TLC, nuSMV, and Lean can be realized as RTL without changing the mathematical model.

The top-level module `drone_closed_loop` stores the current fixed-point state

$$(\hat{q}_1, \hat{q}_2, \hat{q}_3, \hat{v}_1, \hat{v}_2, \hat{v}_3)$$

and implements one closed-loop transition per clock cycle. The module instantiates `drone_controller`, which implements the finite-valued guard-band controller, and `drone_dynamics_update`, which implements the sampled-data update

$$\hat{v}_{k+1} = \hat{v}_k + \frac{\hat{u}_k - Fe_3}{4}, \qquad \hat{q}_{k+1} = \hat{q}_k + \frac{\hat{v}_k}{4} + \frac{\hat{u}_k - Fe_3}{32}.$$

Because the fixed-point scale is $F = 32$ and the sampling period is $\Delta t = 1/4$, these divisions reduce to shifts and additions. Thus, the RTL implementation does not require floating-point arithmetic, multipliers, or DSP blocks.

The Tier 1 RTL implements the box-geofence controller. The Tier 2 RTL keeps the same state registers and dynamics module, but changes the controller and safety monitor according to the nonconvex safe region $S \setminus O$. In particular, the Tier 2 controller adds obstacle-facing guard bands around the no-fly box while preserving the same finite control alphabet. This is important for the project because the change from Tier 1 to Tier 2 is not a change in the physical dynamics; it is a change in the safety specification and in the controller logic required to enforce that specification.

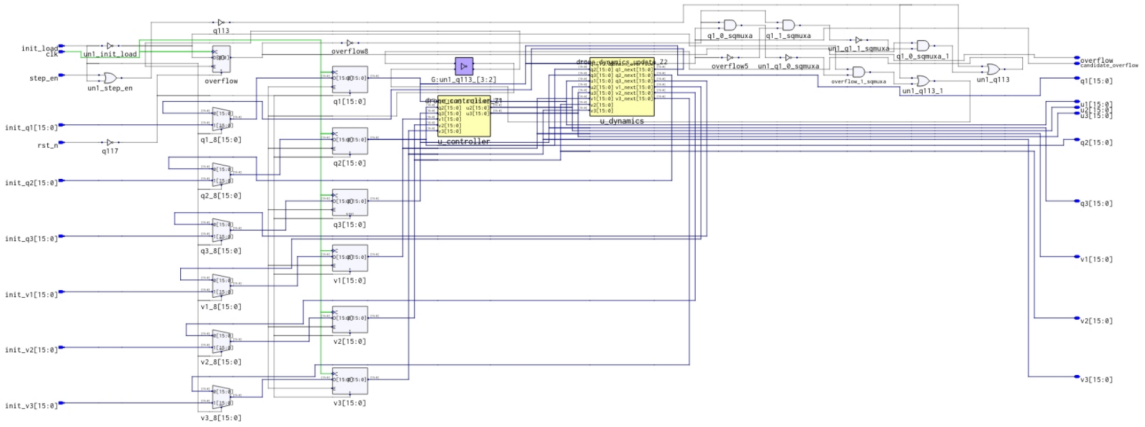


Figure 1: Synthesized RTL schematic for the Tier 1 closed-loop controller. The design stores the fixed-point drone state in registers, computes the guard-band control input, and applies the fixed-point sampled-data update once per clock cycle.

SystemVerilog simulation is used as a finite validation layer. The `drone_safety_monitor` module computes the RTL versions of the geofence predicate, speed predicate, recovery predicate, and, for Tier 2, the obstacle-avoidance predicate. The testbench applies directed tests for boundary behavior and constrained-random tests over admissible fixed-point states. Since simulation only

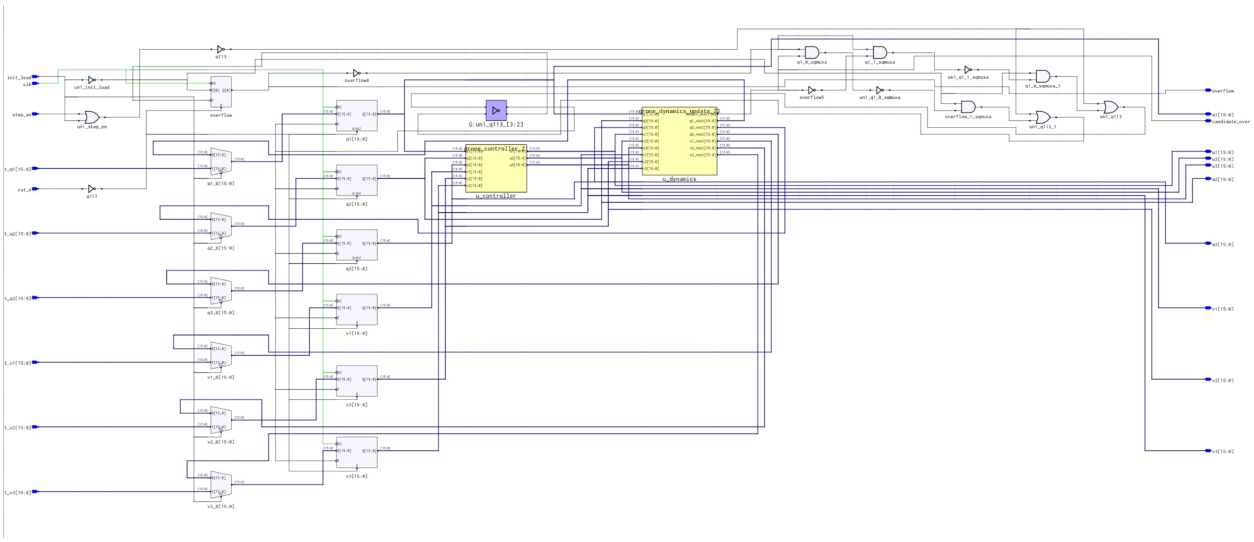


Figure 2: Synthesized RTL schematic for the Tier 2 closed-loop controller. The state registers and dynamics-update datapath remain the same as in Tier 1, while the controller logic expands to include obstacle-avoidance predicates for the nonconvex safe region $S \setminus O$.

checks finitely many executions, it does not replace the model checkers or the Lean proof. Its role is to validate that the RTL artifacts agree with the intended fixed-point transition system and to expose implementation-level mistakes such as incorrect thresholds, sign checks, or obstacle predicates.

The SystemVerilog verification flow was run using Quartus, and the synthesizable RTL was synthesized using Synplify Premier. The implementation schematics are included in this report, while the RTL files, safety monitor, testbench, and simulation script are available in the SystemVerilog repository listed in Appendix.

3 Verification I: TLA+/TLC

3.1 Encoding and Properties

For explicit-state model checking, we encode the Tier 1 sampled system as a finite-state fixed-point transition system in TLA+. Positions, velocities, and controls are represented by scaled integers on a bounded lattice. The controller is encoded directly as a deterministic guard-band transition rule, and an `overflow` flag is used to record any attempted transition outside the chosen finite modeling domain.

The Tier 2 module reuses this fixed-point lattice, sampled-data update, `overflow` handling, and finite control alphabet unchanged. On top of them it adds an open obstacle predicate `InObstacle`, a nonconvex safe-region predicate `SafeRegion` $=$ `Geofence` $\wedge \neg$`InObstacle`, and the augmented controller g_2, which extends the Tier 1 geofence guard band with outward obstacle-avoidance cases gated by a dilated alignment predicate. The admissible initial set is restricted to exclude the obstacle and its dilated braking buffer, and an additional obstacle-avoidance invariant `ObstacleInv` **(P4)** is checked. Because the controller is still a function and no disturbance is introduced, the transition system remains deterministic.

3.2 Specification

The TLA+/TLC specification initializes an admissible encoded state, computes the controller output from the current state, applies the sampled-data update, and marks domain escape explicitly through an `overflow` flag. This structure prevents out-of-domain successors from being silently suppressed: attempted escape is represented as a reachable overflow state and is detected by the `NoOverflow` invariant.

Algorithm 1 TLA+ spec pseudocode

Choose an admissible fixed-point initial state with *overflow* = FALSE
Define finite-domain predicates for type correctness, safe-region membership, inner safety, guard-region membership, and speed safety
Define the guard-band controller using threshold comparisons and velocity sign checks; in Tier 2 also define the obstacle and dilated-alignment predicates
if *overflow* = FALSE **then**
    compute the controller output from the current state
    apply the fixed-point sampled-data update to obtain a candidate successor
    if the candidate successor remains inside the finite TLC domain **then**
        accept the candidate successor
    else
        keep the physical state unchanged and set *overflow* := TRUE
    end if
else
    stutter in the overflow state
end if
Check safety invariants under `Spec`: `TypeOK`, `NoOverflow`, `GeofenceInv`, and `SpeedInv`; in Tier 2 also check `ObstacleInv`
Check recovery under `FairSpec`: `Recovery`

The full TLA+ specification is given in Section B.

3.3 Results

The Tier 1 and Tier 2 TLC result entries are collected in Tables 2, 4 and 5.

4 Verification II: nuSMV

4.1 Encoding

The nuSMV model is a direct transliteration of the TLA+/TLC module onto the same finite fixed-point lattice, so the two arms verify the same discrete system. Position, velocity, controller outputs, and the `overflow` domain flag are finite-domain symbolic variables: positions are integers over the geofence box, velocities lie on the lattice $\{-8, 0, 8\}$ per axis, and the controller outputs are `DEFINE`d as functions of the current state. The sampled-data update is encoded as a relational transition (`TRANS`) over this lattice, using the same scaled integer divisions as the TLC encoding; all divisions are exact for the reachable control alphabet, so no rounding is introduced. As in the TLA+ model, a candidate successor that would leave the finite domain freezes the physical state and latches `overflow`, so an escape attempt becomes an observable reachable state.

Tier 1 uses the baseline geofence guard-band controller. Tier 2 reuses the same update structure and lattice but adds the open obstacle predicate `InObstacle`, the dilated alignment predicates that gate the avoidance cases, and the augmented controller g_2, which adds outward obstacle-braking cases to the Tier 1 geofence band; the admissible initial set additionally excludes the dilated obstacle. Because the controller is deterministic and no disturbance is modeled in Tiers 1–2, the transition relation has exactly one successor per non-overflow state, matching the deterministic interpretation used in the TLA+/TLC section.

4.2 Property Encoding

The safety and liveness properties are stated over the same finite fixed-point state. Each safety property is given both as a symbolic invariant (`INVARSPEC`) and as the equivalent $\square$-formula (`LTLSPEC`): geofence safety **(P1)**, domain containment (`!overflow`, the fixed-point witness of geofence containment), and the speed bound **(P2)**. Recovery **(P3)** is encoded as an LTL eventuality property, requiring that every guard-band state eventually returns to the inner safe set; no fairness constraint is needed because the relation is total and deterministic. In Tier 2 the safe-region predicate `SafeRegion` combines geofence membership with obstacle exclusion ($\texttt{Geofence} \wedge \neg\texttt{InObstacle}$), and obstacle avoidance **(P4)** is additionally checked on its own as both `INVARSPEC` and `LTLSPEC`. All of these guarantees are about the finite symbolic fixed-point model; they make no claim of real-valued universal correctness.

4.3 Results

The Tier 1 and Tier 2 nuSMV result entries are collected in Tables 2 and 4 to 6.

5 Verification III: Lean 4

5.1 Formalization Setup

The Lean formalization follows the mathematical sampled-data model from Section A. It proves safety and invariance obligations deductively for the closed-loop map

$$f(\mathbf{x}) = F(\mathbf{x}, g(\mathbf{x})),$$

where F is the exact sampled-data update and g is the guard-band controller.

The Tier 1 Lean development is organized according to the dependency structure of the model:

$$\texttt{Types} \to \texttt{Dynamics} \to \texttt{Invariants} \to \texttt{Controller} \to \texttt{Properties} \to \texttt{SafetyProofs}.$$

The first files formalize the state space, parameters, sampled dynamics, and geometric safety predicates. The controller and property layers then assemble the closed-loop system and prove basic admissibility results. The final proof layer contains the forward-invariance obligations for the safety goals. Tier 2 adds `Tier2/Obstacle.lean`, `Tier2/Controller.lean`, `Tier2/Properties.lean`, and `Tier2/SafetyProofs.lean` for the obstacle geometry, g_2, and the corresponding safety proofs. The source files are listed in Section D and in the repository.

File	Main contents
`Types.lean`	State and control spaces, parameters, validity assumptions, actuator admissibility.
`Dynamics.lean`	Gravity vector, net acceleration, and exact sampled-data update $F(x, u)$.
`Invariants.lean`	Safe region S, inner safe set S_{in}, and guard region B.
`Controller.lean`	Hover thrust, finite control alphabet, horizontal and vertical guard-band rules, and full controller g.
`Properties.lean`	Closed-loop update $f(x) = F(x, g(x))$, component lemmas, and controller admissibility theorems.
`SafetyProofs.lean`	Forward-invariance statements for `safeRegion` and `innerSafeSet`, and the **P2** velocity-bound invariance (one-step and all iterates).

Table 1: Tier 1 Lean formalization structure.

5.2 Proof Structure

The Lean proofs are one-step invariance arguments plus simple induction where the one-step post-condition reproduces the required hypothesis. For Tier 1, **P2** is proved by iterating `closedLoop_velBound`. For Tier 2 the same structure proves the velocity bound, but the nonconvex safe-region theorem `closedLoopT2_safeRegion'` requires the stronger per-step `speedBound`; consequently the all-iterate safe-region and obstacle-avoidance corollaries keep that assumption explicit.

5.3 Results

The Tier 1 and Tier 2 Lean result entries are collected in Tables 2 and 3.

6 Comparative Analysis

6.1 Property Results

Table 2: Property coverage and verification result by tier and tool.

Tier	Property	TLA+/TLC	nuSMV	Lean 4
1	P1 Geofence safety	No error: `GeofenceInv` under `Spec`.	True: `INVARSPEC SafeRegion` and `LTLSPEC G SafeRegion`.	Proved: `closedLoop_safe`; real-valued and parametric over $\mathbb{R}^3 \times \mathbb{R}^3$.
1	Domain containment	No error: `NoOverflow` under `Spec`.	True: `INVARSPEC !overflow` and `LTLSPEC G !overflow`.	N/A.
1	P2 Speed bound	No error: `SpeedInv` under `Spec`.	True: `INVARSPEC SpeedBound` and `LTLSPEC G SpeedBound`.	Proved: `closedLoop_velBound` and `iterate_velBound`.
1	P3 Recovery	No error: `Recovery` under `FairSpec`.	True: `LTLSPEC` $\Box(B \rightarrow \Diamond S_{\text{in}})$.	Not verified; supporting `innerSafeSet_invariant` requires `driftBound`.
2	P1 Geofence safety $S \setminus O$	No error: Tier 2 `GeofenceInv` under `Spec`.	True: `INVARSPEC SafeRegion` and `LTLSPEC G SafeRegion`.	One-step proved: `closedLoopT2_safeRegion'`; all-iterates statement conditional on `speedBound`.
2	Domain containment	No error: `NoOverflow` under `Spec`.	True: `INVARSPEC !overflow` and `LTLSPEC G !overflow`.	N/A.
2	P2 Speed bound	No error: `SpeedInv` under `Spec`.	True: `INVARSPEC SpeedBound` and `LTLSPEC G SpeedBound`.	Proved: `closedLoopT2_velBound` and `iterate_velBoundT2`.
2	P4 Obstacle avoidance	No error: `ObstacleInv` under `Spec`.	True: `INVARSPEC !InObstacle` and `LTLSPEC G !InObstacle`.	One-step proved through `closedLoopT2_safeRegion'`; all-iterates corollary `closedLoopT2_obstacle_under_speedBound` is conditional on `speedBound`.
2	P3 Recovery	No error: `Recovery` under `FairSpec`.	True: `LTLSPEC` $\Box(B \rightarrow \Diamond S_{\text{in}})$.	Not verified; `closedLoopT2_recovery` and `obstacleBuffer_recovery` are explicit `sorry` placeholders.

Table 3: Lean proof-status data reported by tier.

Tier	Checked corpus	Scope	Axioms / placeholders	Controller result
1	`sorry`-free; proofs mechanized with Harmonic's Aristotle (Achim and and 21 other authors, 2025).	Real-valued, parametric over $\mathbb{R}^3 \times \mathbb{R}^3$ for every `Params.Valid` instance, with theorem-statement hypotheses.	`#print axioms`: `propext`, `Classical.choice`, `Quot.sound`; P3 not proved.	`controller_admissible`.
2	Safety and velocity theorems are `sorry`-free; proofs mechanized with Harmonic's Aristotle.	All parameters and obstacle configurations satisfying `Params.Valid` and `Obstacle.Valid`, with explicit speed and timestep hypotheses.	P3 placeholders: `closedLoopT2_recovery`, `obstacleBuffer_recovery`.	`controllerT2_admissible`.

Verification coverage map. Tables 2 and 3 compare the tools property by property, but the main pattern is easier to see as a coverage map. Figure 3 separates three kinds of positive results: finite-state verification in the model checkers, completed real-valued proof in Lean, and conditional or one-step Lean statements. The figure shows that TLC and nuSMV cover all checked Tier 1 and Tier 2 properties on the fixed-point model, while Lean gives stronger mathematical guarantees for the safety and velocity obligations that have been fully mechanized. The remaining gap is concentrated in recovery: $P3$ is verified by both model checkers on the finite transition system, but it is not fully discharged in the Lean development.

6.2 State Space and Solver Diagnostics

Table 4: Search, reachability, and BDD diagnostics collected from the model-checking runs.

Tool	Tier	Run	Declared states	Generated	Reachable / distinct	Initial	Diameter	Outdegree	BDD init	BDD trans.	BDD/ADD final
TLA+/TLC	1	Safety	N/A	6,885,998	3,709,475	3,176,523	8	1	N/A	N/A	N/A
TLA+/TLC	1	Recovery	N/A	6,885,998	3,709,475	3,176,523	8	1	N/A	N/A	N/A
TLA+/TLC	2	Safety	N/A	5,156,012	2,949,788	2,206,224	8	1	N/A	N/A	N/A
TLA+/TLC	2	Recovery	N/A	5,156,012	2,949,788	2,206,224	8	1	N/A	N/A	N/A
nuSMV	1	Full file	$2^{23.82} \approx 1.48 \times 10^7$	N/A	$2^{21.82} \approx 3.71 \times 10^6$	N/A	8	N/A	29	861	473 771
nuSMV	2	Full file	$2^{23.82} \approx 1.48 \times 10^7$	N/A	$2^{21.49} \approx 2.95 \times 10^6$	N/A	8	N/A	54	$1{,}415 + 256$	287 032
Lean 4	1	Proof	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Lean 4	2	Proof	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

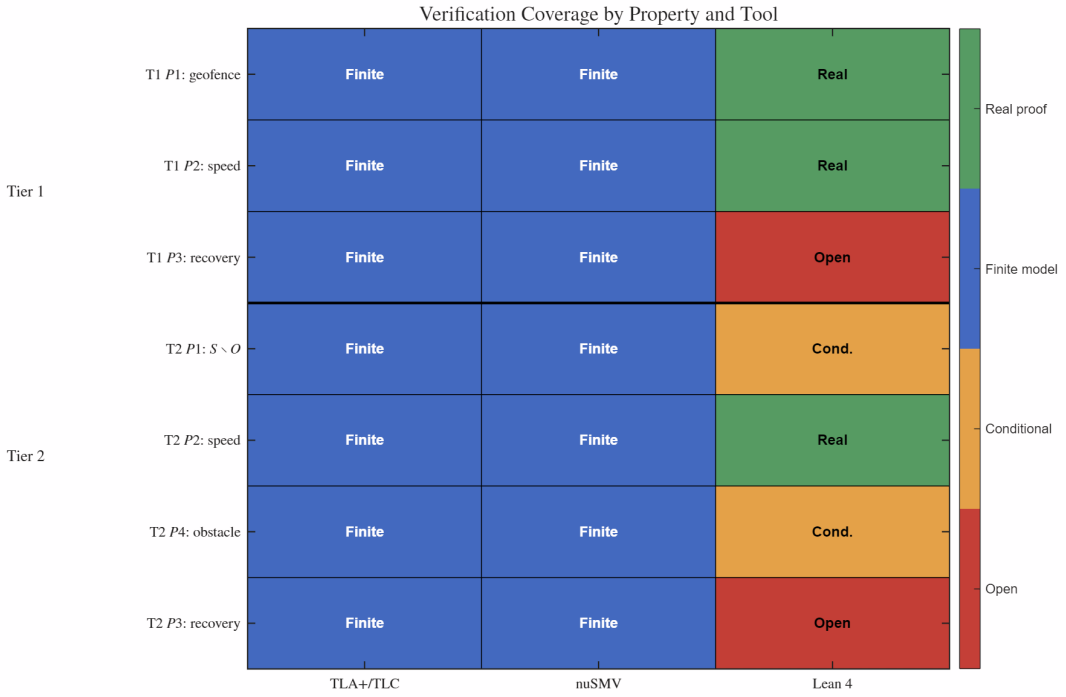


Figure 3: Verification coverage and guarantee type across the main properties. "Finite" denotes exhaustive verification of the fixed-point transition system by a model checker. "Real" denotes a completed Lean theorem over the real-valued sampled-data model. "Cond." denotes a conditional or one-step Lean result whose all-iterate form still depends on explicit hypotheses. "Open" denotes an unproven proof obligation. The main unresolved gap is recovery $P3$, which is model-checked successfully but remains incomplete in Lean.

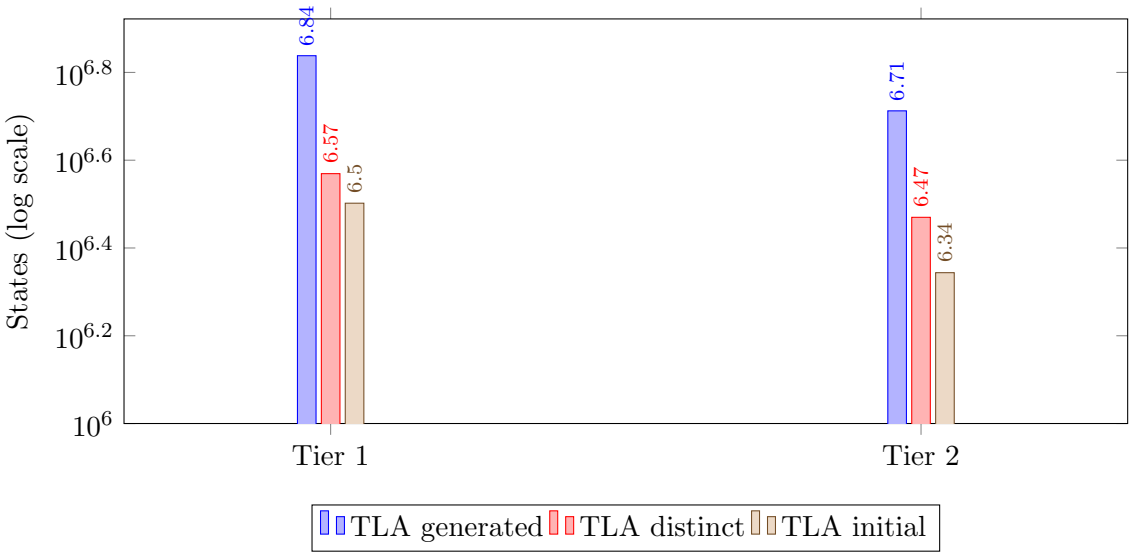


Figure 4: TLA+/TLC state counts from the raw search-statistics tables. Safety and recovery use the same counts in each tier.

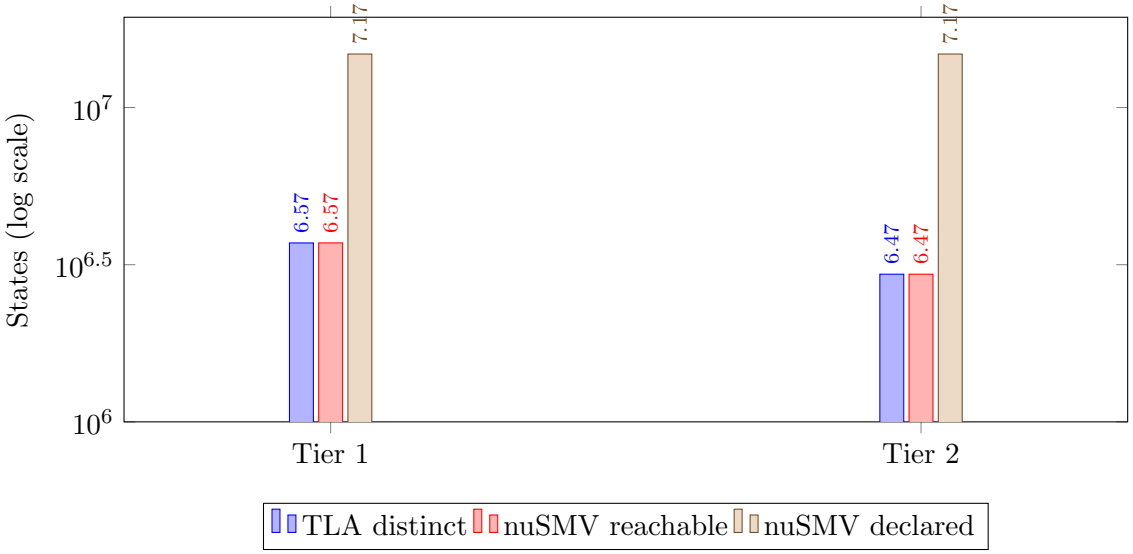


Figure 5: Reachable-state comparison for the shared finite model. nuSMV reachable and declared-state values use the approximations reported in the raw nuSMV tables.

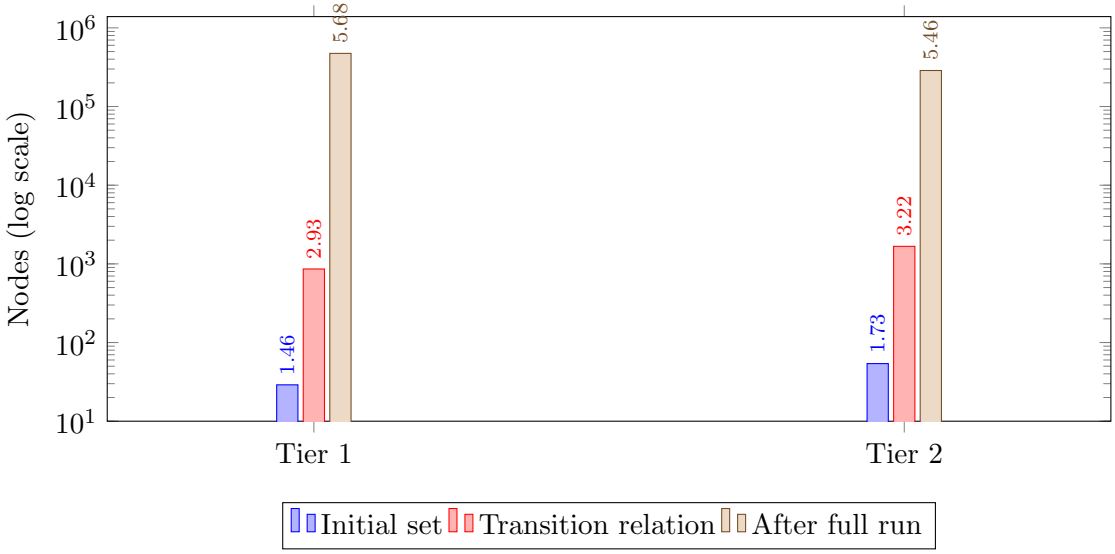


Figure 6: nuSMV BDD/ADD diagnostics from the raw state-space tables. The Tier 2 transition-relation value plots the recorded two-cluster total $1{,}415 + 256$.

6.3 Runtime

Table 5: Recorded verification runtimes.

Tier	TLA safety	TLA recovery	TLA temporal check	nuSMV full file	nuSMV per-property	Lean runtime
Tier 1	28 s	2 min 08 s	39 s	1.46 s	N/A	N/A
Tier 2	35 s	2 min 39 s	40 s	5.66 s	N/A	N/A

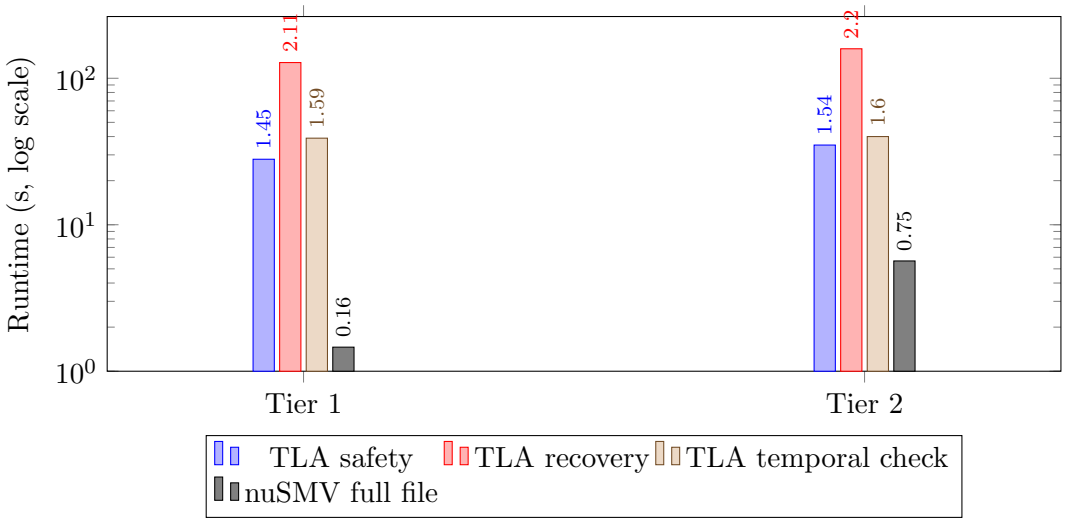


Figure 7: Recorded model-checking runtimes using the data in Table 5. Lean runtime is N/A.

6.4 nuSMV Counterexample Check

Table 6: Recorded Tier 2 nuSMV non-vacuity check for the obstacle dilation.

Item	**Recorded value**
Changed guard	Alignment predicate computed on the bare obstacle slab $[\hat{o}_j^{\min}, \hat{o}_j^{\max}]$ instead of the dilated slab $[\hat{o}_j^{\min} - \hat{\delta}_O, \hat{o}_j^{\max} + \hat{\delta}_O]$.
nuSMV result	**P1** and **P4** false.
Counterexample initial state	$\hat{\mathbf{q}} = (18, 18, 32)$, $\hat{\mathbf{v}} = (-8, -8, 0)$.
Counterexample obstacle state	$\hat{\mathbf{q}} = (7, 7, 32) \in \hat{O}$.
TLA+/TLC value	N/A.
Lean 4 value	N/A.

Visualization of the obstacle-dilation counterexample. The non-vacuity check in Table 6 is easiest to interpret geometrically. With the bare alignment predicate, the diagonal approach to a corner of $\hat{O}$ is not detected early enough. The controller begins braking only when the state has already reached the bare obstacle slab, and the sampled update still contains the current inward velocity. As a result, the next successor enters the open obstacle. With the dilated alignment predicate, the same approach is detected while the state is still in the obstacle buffer, so the corrective input arrests the inward velocity before the open obstacle is reached.

This figure also explains why the obstacle dilation is not merely a conservative modeling choice. Since the obstacle is open, touching a face such as $\hat{q}_1 = 8$ or $\hat{q}_2 = 8$ is still safe, but the next sampled update can enter $\hat{O}$ if braking begins too late. The dilated alignment predicate moves the triggering condition outward from the bare obstacle slab, which gives the controller enough discrete-time braking margin to maintain $P4$.

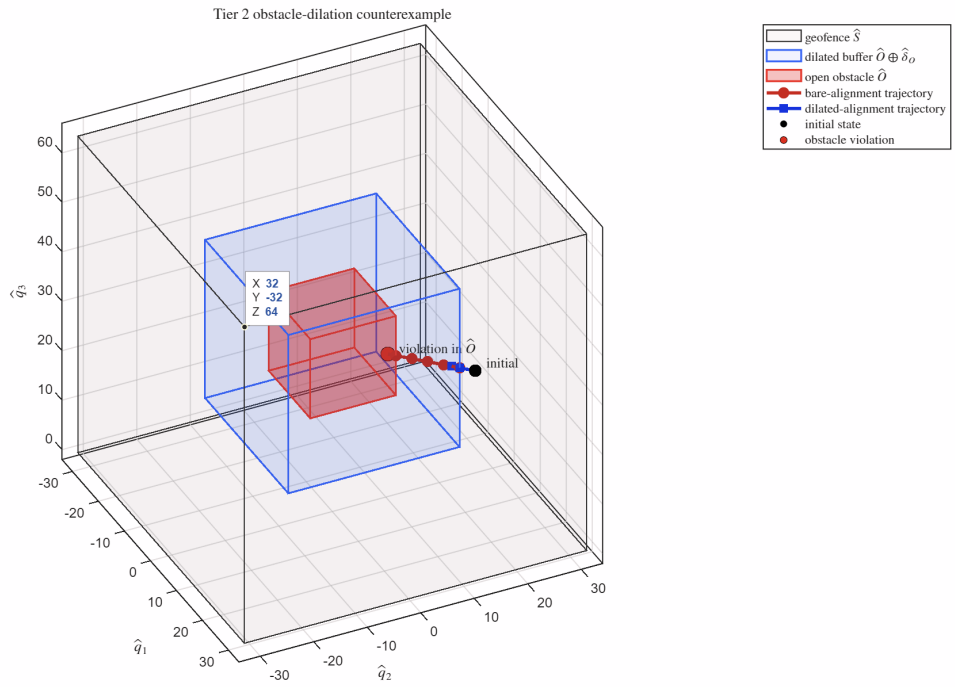


Figure 8: Geometric interpretation of the Tier 2 obstacle-dilation non-vacuity check. The red trajectory shows the diagonal counterexample that appears when obstacle alignment is computed only on the bare obstacle slab. Starting from $\hat{q} = (18, 18, 32)$ and $\hat{v} = (-8, -8, 0)$, the controller waits until the state is already at the corner of the bare slab and the next sampled update lands at $\hat{q} = (7, 7, 32) \in \hat{O}$. The blue trajectory shows the corresponding behavior with the dilated alignment predicate; braking activates on the expanded slab and the state remains outside the open obstacle.

6.5 Summary of Analysis Results

The main result of the comparison is that the finite fixed-point model is verified by both model-checking backends at both complexity tiers. In Tier 1, TLA+/TLC reports no error for geofence safety, domain containment, speed boundedness, and recovery under the fairness-augmented specification, while nuSMV reports the corresponding invariant and LTL encodings true. In Tier 2, the same pattern holds after replacing the convex geofence predicate with the nonconvex predicate $S \setminus O$ and adding the obstacle-avoidance invariant $P4$. Thus, for the finite transition system, the model checkers agree on both the baseline box-geofence system and the obstacle-avoidance extension.

The Lean results give a different kind of guarantee. Instead of enumerating the fixed lattice, Lean formalizes the real-valued sampled-data dynamics and proves safety obligations under explicit mathematical hypotheses. For Tier 1, geofence safety and velocity boundedness are mechanized, including the iterated velocity-bound theorem. For Tier 2, the velocity-bound theorem remains mechanized and obstacle avoidance follows through the one-step nonconvex safe-region theorem, with all-iterate statements requiring the explicit per-step speedBound hypothesis. The main gap is liveness: the recovery property $P3$ is verified by TLA+/TLC and nuSMV on the finite model, but it is not fully proved in Lean. In particular, the Tier 2 recovery statements remain as explicit placeholders. This is the central guarantee-strength tradeoff shown by Tables 2 and 3.

The state-space diagnostics show that the explicit and symbolic encodings are checking the

same reachable finite system. TLA+/TLC reaches 3,709,475 distinct states in Tier 1 and 2,949,788 distinct states in Tier 2, and nuSMV reports the same reachable counts up to its rounded powers-of-two notation. In both tiers the reachability diameter is 8, and the TLA+/TLC outdegree is 1, which agrees with the deterministic closed-loop controller. Tier 2 has fewer reachable states, not because the dynamics are simpler, but because the admissible initial set excludes the dilated obstacle buffer, and the obstacle-invariant property prevents entry into O.

The runtime and BDD data show the expected benefit and cost of symbolic representation. nuSMV is much faster on the finite models, completing the full files in 1.46 s for Tier 1 and 5.66 s for Tier 2, while the TLA+/TLC safety checks take 28 s and 35 s, and the recovery checks take 2 min 08 s and 2 min 39 s. At the same time, the Tier 2 transition relation is symbolically larger in nuSMV, increasing from 861 BDD nodes to the recorded two-cluster value $1,415 + 256$, even though the final BDD/ADD count after the full run is lower. This shows that symbolic difficulty is not determined only by the number of reachable states; it also depends on the Boolean structure of the predicates, especially the obstacle-alignment and nonconvex safety logic.

Finally, Table 6 gives an important debugging result. When the obstacle alignment predicate is weakened from the dilated slab to the bare obstacle slab, nuSMV returns $P1$ and $P4$ false and finds the diagonal approach from $\hat{q} = (18, 18, 32)$, $\hat{v} = (-8, -8, 0)$ to $\hat{q} = (7, 7, 32) \in \hat{O}$. This counterexample shows that the dilated obstacle buffer is load-bearing rather than cosmetic. It also illustrates a useful advantage of model checking: when a geometric controller condition is too weak, the tool returns an explicit failing execution that directly informs the controller design.

7 Conclusion

This project compared deductive verification and model checking for the physical safety of a feedback-controlled drone. We defined a common sampled-data closed-loop model, encoded it on a finite fixed-point lattice for TLA+/TLC and nuSMV, and formalized the corresponding real-valued dynamics and safety predicates in Lean 4. Tier 1 verified the baseline geofence controller, while Tier 2 added a nonconvex no-fly obstacle and an augmented guard-band controller. Across these tiers, the three verification arms exercised the same core controller idea from complementary perspectives: explicit reachable-state enumeration, symbolic state-set manipulation, and deductive proof over mathematical definitions.

The most important empirical finding is agreement between the explicit and symbolic finite-state models. TLC and nuSMV both verified geofence safety, domain containment, speed boundedness, recovery from the guard region, and, in Tier 2, obstacle avoidance. They also produced matching reachability diameters and matching reachable-state counts, which gives confidence that the two encodings represent the same fixed-point transition system. However, their performance profiles were different. TLC gave direct reachable-state diagnostics and interpretable finite traces, but ran slower, especially for recovery. nuSMV used symbolic BDD/ADD representations and completed the full checks in seconds, but its internal complexity was less transparent and depended on the Boolean structure of the encoded predicates.

The Lean results show why deductive verification remains valuable even when model checking is faster on the finite instances. The Lean theorems are not limited to one enumerated fixed-point lattice in the same way as TLC and nuSMV; they state safety properties over the real-valued sampled system under explicit validity hypotheses. This makes the guarantee stronger and more parametric, but the proof effort is higher. In particular, safety and velocity obligations were mechanized, whereas the recovery property remains the major unfinished proof obligation. Thus, the comparison does not identify a single best tool. It shows that model checking is effective for

exhaustive finite-state validation and counterexample generation, while Lean is better suited for reusable mathematical guarantees once the needed invariance and progress lemmas are available.

The SystemVerilog implementation closes the loop between specification and implementation. The controller is finite-valued and threshold-based, and the fixed-point update at $F = 32$ and $\Delta t = 1/4$ reduces to shifts and additions. The RTL synthesis and simulation artifacts therefore support the central modeling decision: the transition system verified in TLA+/TLC and nuSMV is not merely a mathematical abstraction, but can be mapped into hardware-oriented logic without changing the controller structure. Since simulation is finite, it is not a replacement for the formal results, but it is useful for exposing implementation-level mistakes in thresholds, signs, and obstacle predicates.

The main limitations are the finite fixed-point scope of the model checkers, the absence of disturbances and richer continuous dynamics, and the incomplete Lean proof of liveness. Future work should first complete the Lean proof of $P3$, including an obstacle-aware recovery theorem for a Tier 2 inner safe set such as $S_{\text{in}}^{(2)} = S_{\text{in}} \setminus O_{\delta_O}$. A second direction is to prove a refinement connection between the real-valued Lean model, the fixed-point transition systems, and the RTL implementation. Finally, adding disturbances, sensor error, and more complex obstacles would test whether the same guard-band design methodology continues to scale beyond the deterministic model studied here.

Overall, our results support a layered verification workflow for hardware-oriented cyber-physical controllers. Model checking gives fast finite-state assurance and concrete counterexamples during design. Deductive verification gives stronger mathematical statements about the closed-loop dynamics. RTL simulation and synthesis check that the verified transition system can be implemented using realistic hardware logic. Used together, these methods provide a more complete safety argument than any one of them provides in isolation.

A Mathematical Model

A.1 Continuous Dynamics

We model the drone as a point mass under gravity and an external thrust generated by our controller in $\mathbb{R}^3$. For simplicity, we normalize $m = g = 1$. This preserves the form of the dynamics while simplifying the analysis. Physical units can be recovered by the appropriate rescaling of time and thrust.

Definition 1 (Configuration space)**.** The configuration space is

$$Q = \mathbb{R}^3,$$

with configuration coordinates

$$\mathbf{q} = (q_1, q_2, q_3) \in Q,$$

where q_3 denotes altitude.

Definition 2 (State space)**.** The state space is the tangent bundle

$$X = TQ \cong \mathbb{R}^3 \times \mathbb{R}^3 \cong \mathbb{R}^6.$$

A state is written as

$$\mathbf{x} = (\mathbf{q}, \mathbf{v}) \in X,$$

where $\mathbf{v} := \dot{\mathbf{q}} \in \mathbb{R}^3$.

Definition 3 (Control space)**.** The control input is

$$\mathbf{u} = (u_1, u_2, u_3) \in \mathbb{R}^3,$$

We impose axis-wise actuator bounds

$$U = [-u_h, u_h] \times [-u_h, u_h] \times [u_3^{\min}, u_3^{\max}],$$

where $u_h > 0$, and $u_3^{\min}, u_3^{\max}$ satisfy

$$0 \le u_3^{\min} < 1 < u_3^{\max}.$$

Lagrangian. The kinetic and potentials are

$$T = \frac{1}{2}\|\dot{\mathbf{q}}\|^2 = \frac{1}{2}(\dot{q}_1^2 + \dot{q}_2^2 + \dot{q}_3^2), \qquad V = q_3.$$

Hence the Lagrangian is

$$L(\mathbf{q}, \dot{\mathbf{q}}) = T - V = \frac{1}{2}(\dot{q}_1^2 + \dot{q}_2^2 + \dot{q}_3^2) - q_3.$$

Equations of motion. Treating the control input $\mathbf{u}$ as an external generalized force, the forced Euler–Lagrange equations are

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = u_i, \qquad i = 1, 2, 3.$$

Evaluating component-wise, and let $\mathbf{e}_3 = (0, 0, 1)$

$$\begin{aligned} \ddot{q}_1 &= u_1, \\ \ddot{q}_2 &= u_2, \\ \ddot{q}_3 &= u_3 - 1. \end{aligned}$$

In vector form we have

$$\ddot{\mathbf{q}} = \mathbf{u} - \mathbf{e}_3. \tag{7}$$

First-order form. We can also express the dynamics as a first-order system on TQ:

$$\dot{\mathbf{q}} = \mathbf{v}, \qquad \dot{\mathbf{v}} = \mathbf{u} - \mathbf{e}_3. \tag{8}$$

The dynamics are linear in the state space and affine in the control space, with a constant drift term due to gravity.

A.2 Discretization

To obtain a discrete transition system suitable for model checking and inductive reasoning, we sample the continuous dynamics at a fixed period $\Delta t > 0$.

We impose a zero-order-hold assumption (Boyd, n.d.) on the control input:

$$\mathbf{u}(t) = \mathbf{u}_k, \qquad k\Delta t \le t < (k+1)\Delta t, \quad \forall k \in \mathbb{N}.$$

Under this assumption, the continuous dynamics

$$\dot{\mathbf{q}} = \mathbf{v}, \qquad \dot{\mathbf{v}} = \mathbf{u} - \mathbf{e}_3$$

can be integrated exactly on each sampling interval. Indeed, on $[t_k, t_{k+1})$ one has $\mathbf{u}(t) = \mathbf{u}_k$, so $\dot{\mathbf{v}} = \mathbf{u}_k - \mathbf{e}_3$ is constant. Integrating first $\dot{\mathbf{v}}$ and then $\dot{\mathbf{q}} = \mathbf{v}$ yields the sampled-data update below.

Definition 4 (Sampled-data update)**.** Given a state $\mathbf{x}_k = (\mathbf{q}_k, \mathbf{v}_k) \in X$ at time $t_k = k\Delta t$ and a control input $\mathbf{u}_k \in U$ held constant on $[t_k, t_{k+1})$, the discrete-time update map $F : X \times U \to X$ is

$$F(\mathbf{x}_k, \mathbf{u}_k) = \begin{pmatrix} \mathbf{q}_k + \Delta t\, \mathbf{v}_k + \frac{1}{2}\Delta t^2(\mathbf{u}_k - \mathbf{e}_3) \\ \mathbf{v}_k + \Delta t(\mathbf{u}_k - \mathbf{e}_3) \end{pmatrix} = \begin{pmatrix} \mathbf{q}_{k+1} \\ \mathbf{v}_{k+1} \end{pmatrix}. \tag{9}$$

Equivalently,

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \Delta t\, \mathbf{v}_k + \frac{1}{2}\Delta t^2(\mathbf{u}_k - \mathbf{e}_3), \tag{10}$$

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \Delta t(\mathbf{u}_k - \mathbf{e}_3). \tag{11}$$

(9) describes the open-loop dynamics. Once a controller $g : X \to U$ is defined (Section A.3), the closed-loop autonomous system is

$$f(\mathbf{x}_k) = F(\mathbf{x}_k, g(\mathbf{x}_k)),$$

which is the discrete dynamical system (X, f) that we verify.

Remark 1 (Sampling period). Our discrete model is the exact sampled-data system induced by the continuous dynamics under piecewise-constant control. In particular, the one-step update introduces no numerical integration error of the kind arising from approximation schemes such as Forward Euler or Runge–Kutta. The parameter Δt determines the controller's sampling rate. In the finite TLA+/TLC, nuSMV, and RTL encodings used for the experiments, we instantiate $\Delta t = 1/4$; in the Lean development, timestep-dependent obligations are kept explicit as theorem hypotheses.

A.3 Controller Design

We define our deterministic state-feedback controller

$$g : X \to U,$$

as a map that given a state $\mathbf{x} = (\mathbf{q}, \mathbf{v}) \in X$, maps to a control input $\mathbf{u} = (u_1, u_2, u_3) \in U$ which applies corrective thrust when the drone is near the geofence and moving outward, and it applies normalized hover input in the upward direction and zero thrust in the horizontal directions when away from the geofence.

Definition 5 (Geofence)**.** The geofenced safe region is the axis-aligned box

$$S = [q_1^{\min}, q_1^{\max}] \times [q_2^{\min}, q_2^{\max}] \times [q_3^{\min}, q_3^{\max}] \subset \mathbb{R}^3.$$

Definition 6 (Inner safe set and guard region)**.** Fix guard-band widths

$$\delta_h > 0, \qquad \delta_3^- > 0, \qquad \delta_3^+ > 0.$$

Define the inner safe set

$$S_{\text{in}} := [q_1^{\min} + \delta_h,\ q_1^{\max} - \delta_h] \times [q_2^{\min} + \delta_h,\ q_2^{\max} - \delta_h] \times [q_3^{\min} + \delta_3^-,\ q_3^{\max} - \delta_3^+].$$

The guard region is the boundary

$$B := S \setminus S_{\text{in}}.$$

Definition 7 (Finite control alphabet)**.** Define

$$U_{\text{ctrl}} := \left\{ \mathbf{u} = (u_1, u_2, u_3) \in U : u_1, u_2 \in \{-u_h, 0, +u_h\},\ u_3 \in \{u_3^{\min}, u_{\text{hov}}, u_3^{\max}\} \right\}.$$

where $u_{\text{hov}} = 1$ is the normalized hover thrust, $u_3^{\max} > 1$ is the corrective upward thrust, and $u_3^{\min} < 1$ is the corrective downward thrust.

Definition 8 (Guard-band recovery controller)**.** For a state $\mathbf{x} = (\mathbf{q}, \mathbf{v}) \in X$, the control $g(\mathbf{q}, \mathbf{v}) = \mathbf{u} = (u_1, u_2, u_3)$ is defined componentwise as follows.

Horizontal axes ($i = 1, 2$):

$$u_i(q, v) = \begin{cases} +u_h & \text{if } q_i \le q_i^{\min} + \delta_h \text{ and } v_i < 0, \\ -u_h & \text{if } q_i \ge q_i^{\max} - \delta_h \text{ and } v_i > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Vertical axis:

$$u_3(q, v) = \begin{cases} u_3^{\max} & \text{if } q_3 \le q_3^{\min} + \delta_3^- \text{ and } v_3 < 0, \\ u_3^{\min} & \text{if } q_3 \ge q_3^{\max} - \delta_3^+ \text{ and } v_3 > 0, \\ u_{\text{hov}} & \text{otherwise.} \end{cases}$$

Proposition 1. *The guard-band controller $g : X \to U$ is well-defined and satisfies*

$$g(x) \in U_{\text{ctrl}} \subseteq U, \quad \forall x \in X.$$

Proof. For each $\mathbf{x} = (\mathbf{q}, \mathbf{v}) \in X$, g assigns to each horizontal component a value in $\{-u_h, 0, +u_h\}$, and assigns to the vertical component a value in $\{u_3^{\min}, u_{\text{hov}}, u_3^{\max}\}$. Hence $g(x) \in U_{\text{ctrl}}$. Since $U_{\text{ctrl}} \subseteq U$, it follows that $g(x) \in U$. □

Thus, the controller applies corrective thrust only when the state lies in the guard region B and the corresponding velocity component points outward. When the state is away from the guard region, the controller applies the normalized input

$$(0, 0, u_{\text{hov}}).$$

This controller design follows the guideline outlined in Lygeros et al. (1999) for designing the least restrictive controllers that guarantees safety while leaving freedom for other objectives such as efficiency and performance. We identified an invariant safe region, then restrict the controller near the boundary to inputs that prevent escape.

Remark 2 (**Guard-band controller is hardware-oriented**). The map g is memoryless, finite-valued, and satisfies

$$g(x) \in U_{\text{ctrl}} \subseteq U, \quad \forall x \in X,$$

it is nonblocking and hardware-oriented. Its implementation requires only threshold comparisons, sign checks, and selection from a small finite alphabet, which makes it suitable for RTL-style logic.

A.4 Hardware-Oriented Controller

The guard-band controller is intentionally chosen to be hardware-oriented. Both the Tier 1 controller g and the Tier 2 obstacle-avoidance controller g_2 are memoryless, deterministic, and finite-valued. Their outputs are selected from the finite alphabet

$$U_{\text{ctrl}} = \{-u_h, 0, +u_h\} \times \{-u_h, 0, +u_h\} \times \{u_3^{\min}, u_{\text{hov}}, u_3^{\max}\}.$$

Thus, implementing the controller does not require floating-point arithmetic, online optimization, or a memory-dependent policy. The controller can be realized using only signed threshold comparisons, velocity sign checks, Boolean combinations of guard predicates, and multiplexing among finitely many control values.

This structure is the reason the mathematical model maps directly to the SystemVerilog RTL. The RTL stores the fixed-point state

$$(\hat{q}_1, \hat{q}_2, \hat{q}_3, \hat{v}_1, \hat{v}_2, \hat{v}_3)$$

in registers and applies one closed-loop transition per clock cycle. Positions, velocities, and controls are represented as scaled integers with scale factor

$$F = 32,$$

matching the fixed-point encoding used in the TLA+/TLC and nuSMV models. For the sampling period

$$\Delta t = \frac{1}{4},$$

the sampled-data dynamics become

$$\hat{v}_{k+1} = \hat{v}_k + \frac{\hat{u}_k - Fe_3}{4}, \qquad \hat{q}_{k+1} = \hat{q}_k + \frac{\hat{v}_k}{4} + \frac{\hat{u}_k - Fe_3}{32}.$$

Because the divisions are by powers of two, the synthesized closed-loop controller/dynamics datapath reduces to shifts, additions, subtractions, signed threshold comparisons, and multiplexing

among finitely many control values. It therefore does not require floating-point arithmetic, dividers, multipliers, or DSP blocks in the main closed-loop datapath. The safety monitor is simulation/checking instrumentation rather than the synthesized main design.

The Tier 1 RTL, as described in Section 2.4, implements the box-geofence guard-band controller. The Tier 2 RTL bundle keeps the same fixed-point state registers and sampled-data update module, but extends the controller and safety monitor with the obstacle predicates used for the nonconvex safe region $S \setminus O$. In particular, the Tier 2 controller adds obstacle-facing guard-band cases gated by the dilated alignment predicate. Thus, the transition from Tier 1 to Tier 2 changes the safety predicate and Boolean controller logic, not the sampled physical dynamics.

A.5 Safety and Liveness Properties

Let $X_{\text{init}} \subseteq X$ denote the set of admissible initial states. For each initial state

$$\mathbf{x}^{\text{init}} \in X_{\text{init}},$$

the closed-loop map f generates the forward orbit

$$\mathcal{O}^+(\mathbf{x}^{\text{init}}) = \{\mathbf{x}^{\text{init}},\, f(\mathbf{x}^{\text{init}}),\, f^2(\mathbf{x}^{\text{init}}),\, \dots\}.$$

Writing

$$\mathbf{x}_k := f^k(\mathbf{x}^{\text{init}}), \qquad k \in \mathbb{N},$$

we obtain the discrete-time execution

$$\mathbf{x}^{\text{init}}, \mathbf{x}_1, \mathbf{x}_2, \dots, \qquad \mathbf{x}_{k+1} = f(\mathbf{x}_k).$$

All temporal properties are stated in linear temporal logic (LTL) over the infinite execution

$$\sigma = (\mathbf{x}_k)_{k \geq 0},$$

following the standard syntax of (Pnueli, 1977). Let $\mathcal{AP}$ denote a set of atomic propositions. LTL formulas are generated by the BNF grammar:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid X\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2, \qquad p \in \mathcal{AP}.$$

We use the derived temporal operators

$$\Diamond\varphi := \top \,\mathcal{U}\, \varphi, \qquad \Box\varphi := \neg\Diamond\neg\varphi.$$

Definition 9 (Safety and Liveness Properties)**.**

P1. **Geofence safety.** The drone always remains in the geofenced region:

$$\Box\,(\mathbf{q} \in S).$$

P2. **Speed bound.** The drone never exceeds the prescribed speed limit:

$$\Box\,(\|\mathbf{v}\| \leq v_{\max}).$$

P3. **Recovery from the guard region.** Whenever the drone enters the guard region, it eventually returns to the inner safe region:

$$\Box\,(\mathbf{q} \in B \rightarrow \Diamond\,(\mathbf{q} \in S_{\text{in}})).$$

Remark 3. Properties P1 and P2 are safety properties. Property P3 is a liveness property.

B Model Checkers Encoding

B.1 Fixed-Point Encoding

To obtain a finite-state model suitable for TLC and to align with the hardware-oriented interpretation of the controller, we represent the sampled system on a binary fixed-point lattice.

We fix the scale factor

$$F = 32,$$

and encode position, velocity, and control by scaled integer variables

$$\hat{\mathbf{q}} = F\mathbf{q}, \qquad \hat{\mathbf{v}} = F\mathbf{v}, \qquad \hat{\mathbf{u}} = F\mathbf{u}.$$

We also fix the sampling period $\Delta t = \frac{1}{4}$.

For the baseline TLC model, we choose the actuator values

$$u_h = 1, \qquad u_{\text{hov}} = 1, \qquad u_3^{\min} = 0, \qquad u_3^{\max} = 2,$$

so the scaled control alphabet becomes

$$\hat{U}_{\text{ctrl}} = \{-F,\, 0,\, F\} \times \{-F,\, 0,\, F\} \times \{0,\, F,\, 2F\}.$$

Scaled dynamics. Substituting $\hat{\mathbf{q}} = F\mathbf{q}$, $\hat{\mathbf{v}} = F\mathbf{v}$, $\hat{\mathbf{u}} = F\mathbf{u}$, and $\Delta t = \frac{1}{4}$ into the sampled-data update (9) yields the integer dynamics

$$\hat{\mathbf{v}}_{k+1} = \hat{\mathbf{v}}_k + \frac{\hat{\mathbf{u}}_k - F\mathbf{e}_3}{4}, \tag{12}$$

$$\hat{\mathbf{q}}_{k+1} = \hat{\mathbf{q}}_k + \frac{\hat{\mathbf{v}}_k}{4} + \frac{\hat{\mathbf{u}}_k - F\mathbf{e}_3}{32}. \tag{13}$$

Lattice closure. For the horizontal components, the velocity increment is

$$\hat{u}_i/4 \in \{-8,\, 0,\, 8\}, \qquad i = 1, 2.$$

For the vertical component, the increment is

$$(\hat{u}_3 - F)/4 \in \{-8,\, 0,\, 8\}.$$

Hence if $\hat{\mathbf{v}}_k \in (8\mathbb{Z})^3$, then $\hat{\mathbf{v}}_{k+1} \in (8\mathbb{Z})^3$. Moreover, $\hat{\mathbf{v}}_k/4 \in (2\mathbb{Z})^3$, while each component of

$$\frac{\hat{\mathbf{u}}_k - F\mathbf{e}_3}{32}$$

lies in $\{-1,\, 0,\, 1\}$. Therefore $\hat{\mathbf{q}}_{k+1} \in \mathbb{Z}^3$ whenever $\hat{\mathbf{q}}_k \in \mathbb{Z}^3$. Since the TLC model uses integer-valued position states, we restrict the initial velocity to the sublattice

$$\hat{\mathbf{v}}_0 \in (8\mathbb{Z})^3,$$

which guarantees that the entire trajectory remains on an integer lattice.

Baseline geofence. We take the geofence bounds

$$\hat{q}_1, \hat{q}_2 \in [-32,\, 32], \qquad \hat{q}_3 \in [0,\, 64],$$

corresponding to $q_1, q_2 \in [-1, 1]$ and $q_3 \in [0, 2]$ in unscaled coordinates. The guard-band widths are

$$\hat{\delta}_h = 8, \qquad \hat{\delta}_3^- = 8, \qquad \hat{\delta}_3^+ = 8,$$

i.e. $\delta_h = \delta_3^- = \delta_3^+ = \frac{1}{4}$, giving horizontal controller thresholds at $\hat{q}_i = -24$ and $\hat{q}_i = 24$ for $i = 1, 2$, and vertical thresholds at $\hat{q}_3 = 8$ and $\hat{q}_3 = 56$.

B.2 Tier 2 Fixed-Point Extensions

The Tier 2 obstacle model (Section C.1) is encoded on the *same* fixed-point lattice as Tier 1: scale $F = 32$, sampling period $\Delta t = \frac{1}{4}$, geofence bounds $\hat{q}_1, \hat{q}_2 \in [-32, 32]$, $\hat{q}_3 \in [0, 64]$, velocity lattice $\hat{V} = \{-8, 0, 8\}^3$, and guard-band widths $\hat{\delta}_h = \hat{\delta}_3^- = \hat{\delta}_3^+ = 8$. All update divisions remain exact for the reachable control alphabet, so the encoding still introduces no rounding. Only the following elements are added.

Obstacle and buffer (scaled). The no-fly box is the *open* box

$$\hat{O} = (-8, 8) \times (-8, 8) \times (24, 40),$$

i.e. $O = (-\frac{1}{4}, \frac{1}{4}) \times (-\frac{1}{4}, \frac{1}{4}) \times (\frac{3}{4}, \frac{5}{4})$ in unscaled coordinates, with buffer width $\hat{\delta}_O = 8$. Each obstacle dimension equals $16 = 2\hat{\delta}_O$, satisfying Assumption 1(i), and the dilation

$$\hat{O} \oplus \hat{\delta}_O = [-16, 16] \times [-16, 16] \times [16, 48]$$

is contained in the inner safe set $\hat{S}_{\text{in}} = [-24, 24] \times [-24, 24] \times [8, 56]$, satisfying Assumption 1(ii). The dilated alignment slab bounds are therefore $\hat{D}_i^{\min} = \hat{o}_i^{\min} - 8$ and $\hat{D}_i^{\max} = \hat{o}_i^{\max} + 8$. Because the per-step inward displacement is at most 2 lattice units and a single corrective step removes a velocity of 8 (settling penetration 1), $\hat{\delta}_O = 8$ discharges the braking-distance condition Assumption 1(iii) with wide margin.

Obstacle membership on the lattice. Since $\hat{O}$ is open, lattice membership is the *strict* predicate

$$\hat{\mathbf{q}} \in \hat{O} \iff \hat{o}_i^{\min} < \hat{q}_i < \hat{o}_i^{\max} \text{ for all } i,$$

so a state coming to rest on a face $\hat{q}_i = \hat{o}_i^{\max}$ is outside $\hat{O}$ and counts as safe.

Avoidance controller. The controller adds, on each axis, the two outward obstacle cases of Section C.1 (gated by the dilated alignment) to the Tier 1 geofence cases. Under Assumption 1 the geofence and obstacle threshold ranges are disjoint, so the encoded controller remains single-valued and finite-valued. Corrective thrust opposes inward motion only, so the velocity increment stays in $\{-8, 0, 8\}$ and the lattice-closure argument of Section B.1 carries over unchanged; the closed-loop transition is still deterministic.

Domain versus safety. Crucially, the obstacle is treated as a *safety property*, not as a modeling-domain wall. The finite domain and the `overflow` latch remain the geofence box and velocity lattice exactly as in Tier 1; since $\hat{O} \subset \hat{S}_{\text{in}}$, a step that lands in $\hat{O}$ is a legal, in-domain, reachable state. Obstacle violation is therefore detected by the invariant `!InObstacle` (property **P4**), not by `overflow`. The admissible initial set additionally excludes the closed dilated obstacle:

$$\hat{X}_{\text{init}}^{(2)} = \big(\hat{S}_{\text{in}} \setminus (\hat{O} \oplus \hat{\delta}_O)\big) \times \hat{V}.$$

B.3 Property Encoding

The safety and liveness properties from Section A.5 are encoded in the TLC model as state predicates and temporal formulas over the fixed-point state variables

$$(\hat{q}_1, \hat{q}_2, \hat{q}_3, \hat{v}_1, \hat{v}_2, \hat{v}_3, \mathit{overflow}).$$

Here $\hat{q}_i$ and $\hat{v}_i$ are integer-valued fixed-point encodings of the sampled position and velocity components, and the Boolean flag *overflow* records whether the trajectory leaves the chosen finite modeling domain.

We first check the invariant

$$\texttt{TypeOK},$$

which asserts that each encoded position component lies in its prescribed integer interval, each encoded velocity component lies in the finite velocity set, and $\mathit{overflow} \in \{\text{FALSE}, \text{TRUE}\}$. This serves as the basic well-formedness condition for the TLC state space.

The geofence property is encoded by the invariant

$$\texttt{GeofenceInv},$$

which requires

$$\hat{q}_1 \in [Q1Min, Q1Max], \qquad \hat{q}_2 \in [Q2Min, Q2Max], \qquad \hat{q}_3 \in [Q3Min, Q3Max].$$

Likewise, the speed bound is encoded by

$$\texttt{SpeedInv},$$

defined by

$$\hat{v}_1^2 + \hat{v}_2^2 + \hat{v}_3^2 \le VMaxSq.$$

This is the fixed-point counterpart of the continuous constraint $\|\mathbf{v}\| \le v_{\max}$.

A separate invariant,

$$\texttt{NoOverflow} \equiv \neg \mathit{overflow},$$

ensures that the trajectory remains within the finite modeling domain. This is important because the specification does not suppress out-of-domain successors; instead, such an escape is made explicit by setting `overflow` to `TRUE`. A failed invariant therefore reflects a genuine domain escape, rather than an artifact of the transition relation.

The recovery property is encoded as the temporal formula

$$\texttt{Recovery} \;=\; \Box\big((\neg \mathit{overflow} \wedge \texttt{BoundaryZone}) \to \Diamond(\neg \mathit{overflow} \wedge \texttt{InnerSafe})\big).$$

Thus, whenever the drone enters the guard region, it must eventually return to the inner safe set, provided the execution remains within the modeled domain.

In TLC, the safety properties are checked first under

$$\texttt{Spec} \;=\; \texttt{Init} \wedge [][\texttt{Next}]_{\texttt{vars}},$$

using `TypeOK`, `NoOverflow`, `GeofenceInv`, and `SpeedInv` as invariants. After these pass, the recovery property is checked under the fairness-augmented specification

$$\texttt{FairSpec} \;=\; \texttt{Init} \wedge [][\texttt{Next}]_{\texttt{vars}} \wedge \mathrm{WF}_{\texttt{vars}}(\texttt{StepNext}),$$

which rules out infinite stuttering and allows TLC to interpret `Recovery` as a genuine liveness requirement.

C Higher-Tier Model Details

This appendix gives the detailed model and controller derivations for Tier 2, extending the Tier 1 development of Section A. The state space, sampled-data update (9), finite control alphabet U_{ctrl}, and properties P1–P3 are inherited unchanged unless stated otherwise.

C.1 Tier 2: Obstacle Geometry and Avoidance Controller

C.1.1 Nonconvex safe region

Definition 10 (Obstacle and safe region)**.** Fix an axis-aligned *open* no-fly box

$$O = (o_1^{\min}, o_1^{\max}) \times (o_2^{\min}, o_2^{\max}) \times (o_3^{\min}, o_3^{\max}) \subset S_{\mathrm{in}}.$$

The Tier 2 safe position region is $S \setminus O$.

Since O is open, obstacle membership uses strict inequalities, and points on ∂O are treated as safe. Equivalently,

$$\mathbf{q} \notin O \iff \bigvee_{i=1}^{3} \big(q_i \le o_i^{\min} \text{ or } q_i \ge o_i^{\max}\big).$$

Unlike the outer geofence condition, which is a conjunction of axis-wise bounds, obstacle avoidance requires keeping at least one obstacle coordinate outside its open interval. The Tier 2 controller enforces this through an obstacle guard band, defined below.

Definition 11 (Dilated alignment and obstacle buffer)**.** Fix a buffer width $\delta_O > 0$. Define the closed δ_O-expanded obstacle box by

$$O_{\delta_O} := [o_1^{\min} - \delta_O, o_1^{\max} + \delta_O] \times [o_2^{\min} - \delta_O, o_2^{\max} + \delta_O] \times [o_3^{\min} - \delta_O, o_3^{\max} + \delta_O].$$

For each axis i, define the orthogonal-alignment predicate

$$\mathrm{aligned}_i(\mathbf{q}) := \bigwedge_{j \neq i} \big(q_j \in [o_j^{\min} - \delta_O,\, o_j^{\max} + \delta_O]\big).$$

The obstacle buffer is the set of safe states lying within distance δ_O of one obstacle face while aligned with the obstacle in the other coordinates:

$$N_{\delta_O}(O) := \big\{\mathbf{q} \in S \setminus O : \exists i,\ \mathrm{aligned}_i(\mathbf{q}) \ \wedge\ q_i \in [o_i^{\max}, o_i^{\max} + \delta_O] \cup [o_i^{\min} - \delta_O, o_i^{\min}]\big\}.$$

The purpose of the dilated alignment condition is to make obstacle avoidance activate before the drone reaches a face, edge, or corner of O. If alignment were checked only against the undilated obstacle slab, then on diagonal approaches the controller could wait until the orthogonal coordinates were already within the obstacle extent. Using the expanded intervals causes the avoidance rule to activate earlier, giving the controller a buffer in which to arrest inward motion before the open obstacle is entered.

Assumption 1 (Obstacle sizing and buffer adequacy)**.**

(i) The obstacle is not degenerate relative to the buffer:

$$o_i^{\max} - o_i^{\min} \geq 2\delta_O \qquad \text{for each } i.$$

(ii) The closed δ_O-expanded obstacle box lies inside the inner safe set:

$$O_{\delta_O} \subseteq S_{\text{in}}.$$

(iii) The buffer width δ_O is chosen to be at least the worst-case sampled braking distance for an inward velocity component of magnitude at most $v_{\max}$, under the available corrective acceleration on the corresponding axis.

Assumption 1(i) is a nondegeneracy condition ensuring that the obstacle is large enough relative to the chosen buffer width. Assumption 1(ii) keeps the obstacle buffer away from the outer geofence guard band, so the obstacle-avoidance logic and geofence-recovery logic do not interfere. Assumption 1(iii) is the braking-margin condition: once the obstacle guard band activates, the available corrective thrust is assumed sufficient to arrest inward motion before the open obstacle is entered.

C.1.2 Avoidance controller

Definition 12 (Tier 2 controller)**.** The Tier 2 controller $g_2 : X \to U$ extends the Tier 1 guard-band controller by adding obstacle-avoidance cases. The geofence cases are unchanged. For horizontal axes $i \in \{1, 2\}$, define

$$u_i(\mathbf{q}, \mathbf{v}) = \begin{cases} +u_h & q_i \leq q_i^{\min} + \delta_h,\ v_i < 0, \\ -u_h & q_i \geq q_i^{\max} - \delta_h,\ v_i > 0, \\ +u_h & \text{aligned}_i(\mathbf{q}),\ q_i \in [o_i^{\max}, o_i^{\max} + \delta_O],\ v_i < 0, \\ -u_h & \text{aligned}_i(\mathbf{q}),\ q_i \in [o_i^{\min} - \delta_O, o_i^{\min}],\ v_i > 0, \\ 0 & \text{otherwise.} \end{cases}$$

For the vertical axis, define

$$u_3(\mathbf{q}, \mathbf{v}) = \begin{cases} u_3^{\max} & q_3 \leq q_3^{\min} + \delta_3^-,\ v_3 < 0, \\ u_3^{\min} & q_3 \geq q_3^{\max} - \delta_3^+,\ v_3 > 0, \\ u_3^{\max} & \text{aligned}_3(\mathbf{q}),\ q_3 \in [o_3^{\max}, o_3^{\max} + \delta_O],\ v_3 < 0, \\ u_3^{\min} & \text{aligned}_3(\mathbf{q}),\ q_3 \in [o_3^{\min} - \delta_O, o_3^{\min}],\ v_3 > 0, \\ u_{\text{hov}} & \text{otherwise.} \end{cases}$$

Then

$$g_2(\mathbf{q}, \mathbf{v}) = (u_1(\mathbf{q}, \mathbf{v}), u_2(\mathbf{q}, \mathbf{v}), u_3(\mathbf{q}, \mathbf{v})).$$

Proposition 2 (Well-definedness)**.** *Assume the inner safe set is nonempty and that Assumption 1 holds. Then the Tier 2 controller g_2 is single-valued and satisfies*

$$g_2(\mathbf{x}) \in U_{\text{ctrl}} \subseteq U, \qquad \forall \mathbf{x} \in X.$$

Proof. Fix an axis i. The two geofence guard bands on that axis are disjoint because the corresponding inner safe interval is nonempty. By Assumption 1(ii), the expanded obstacle box lies inside S_{in}. Hence the obstacle buffer ranges lie inside the inner safe region and are disjoint from the outer geofence guard bands.

The two obstacle-side buffer ranges,

$$[o_i^{\min} - \delta_O, o_i^{\min}] \qquad \text{and} \qquad [o_i^{\max}, o_i^{\max} + \delta_O],$$

are disjoint because $o_i^{\min} < o_i^{\max}$. Therefore, for each axis, at most one guarded position case can apply. The velocity sign conditions only further restrict these cases, so they cannot introduce ambiguity.

Thus each component of g_2 is single-valued. For horizontal axes, the output lies in $\{-u_h, 0, +u_h\}$, and for the vertical axis, the output lies in $\{u_3^{\min}, u_{\text{hov}}, u_3^{\max}\}$. Hence $g_2(\mathbf{x}) \in U_{\text{ctrl}} \subseteq U$ for all $\mathbf{x} \in X$. □

Remark 4 (Corner cases). The obstacle-avoidance rules are applied componentwise, so corrections along different axes can be active at the same time and no tie-break rule is needed. This is useful near edges and corners of O, where the drone may approach the obstacle with several inward velocity components. The dilated alignment condition causes the avoidance rules to activate before the state reaches the bare obstacle slab in the orthogonal coordinates, giving the controller a buffer in which to reduce inward motion. Since O is open, reaching a point on ∂O is still safe; the safety property forbids entering O, not touching its boundary.

With the augmented controller g_2, the Tier 2 closed-loop system is

$$f_2(\mathbf{x}) = F(\mathbf{x}, g_2(\mathbf{x})).$$

Since F is unchanged and g_2 is single-valued, the transition system remains deterministic. The additional property checked in Tier 2 is obstacle avoidance, $\square(\mathbf{q} \notin O)$.

Remark 5 (Scope of recovery at Tier 2). Properties P1 and P2 carry over from Tier 1. The recovery property P3 is also checked in its Tier 1 form,

$$\square\big(\mathbf{q} \in B \to \Diamond(\mathbf{q} \in S_{\text{in}})\big),$$

where $B = S \setminus S_{\text{in}}$ is the outer geofence guard band. Thus, at Tier 2, recovery still means return from the outer guard band to the inner safe set. Since $O \subset S_{\text{in}}$, this property does not express recovery away from the obstacle buffer.

An obstacle-aware recovery property would instead use a Tier 2 inner safe set, for example

$$S_{\text{in}}^{(2)} := S_{\text{in}} \setminus O_{\delta_O},$$

and would require eventual return to $S_{\text{in}}^{(2)}$. We do not verify this stronger liveness property here. Obstacle avoidance is checked separately as the invariant **P4**.

D Lean 4 Formalization

The Lean source is in `lean/DroneLean/DroneLean`. The files used here are:

Tier 1: `Tier1/Types.lean`, `Tier1/Dynamics.lean`, `Tier1/Invariants.lean`
`Tier1/Controller.lean`, `Tier1/Properties.lean`, `Tier1/SafetyProofs.lean`
Tier 2: `Tier2/Obstacle.lean`, `Tier2/Controller.lean`
`Tier2/Properties.lean`, `Tier2/SafetyProofs.lean`

The Tier 2 recovery theorem statements `closedLoopT2_recovery` and `obstacleBuffer_recovery` are present but intentionally left with `sorry`.

E Raw Verification Data

E.1 TLA+/TLC (Tier 1)

TLC checks `DroneTLCV1.tla` with separate safety and recovery configurations. The safety run uses `Spec` and the invariants `TypeOK`, `GeofenceInv`, `NoOverflow`, and `SpeedInv`; the recovery run uses `FairSpec` and the temporal property `Recovery`. Both runs complete with no error.

Both configurations explore 3,709,475 distinct reachable states at depth 8, matching the nuSMV reachable count in Section E.3. The maximum outdegree is 1, as expected for the deterministic closed loop.

Table 7: TLA+/TLC Tier 1 search statistics and per-configuration results.

Metric	**Safety**	**Recovery**
Configuration	`_Safety.cfg`	`_Recovery.cfg`
Specification	`Spec`	`FairSpec`
States generated	6,885,998	6,885,998
Distinct states	3,709,475	3,709,475
Initial states (distinct)	3,176,523	3,176,523
Search depth (diameter)	8	8
Temporal-property check	—	39 s
Result	no error	no error
Wall-clock time	28 s	2 min 08 s

The wall-clock and temporal-check rows are recorded-run diagnostics; the state counts and no-error results were reproduced in a fresh TLC run.

E.2 TLA+/TLC (Tier 2)

TLC checks `DroneTLCV2.tla` with `Spec` for safety and `FairSpec` for recovery. The safety configuration adds `ObstacleInv` ($\neg$`InObstacle`) to the Tier 1 invariants, and `GeofenceInv` denotes the Tier 2 safe region $S \setminus O$. Both runs complete with no error.

Both configurations explore 2,949,788 distinct reachable states at depth 8, again with maximum outdegree 1. This matches the nuSMV reachable count in Section E.4. Tier 2 has fewer reachable states than Tier 1 because the admissible initial set excludes the dilated obstacle buffer, while **P4** rules out entering O.

Table 8: TLA+/TLC Tier 2 search statistics and per-configuration results.

Metric	**Safety**	**Recovery**
Configuration	`_Safety.cfg`	`_Recovery.cfg`
Specification	`Spec`	`FairSpec`
States generated	5,156,012	5,156,012
Distinct states	2,949,788	2,949,788
Initial states (distinct)	2,206,224	2,206,224
Search depth (diameter)	8	8
Temporal-property check	—	40 s
Result	no error	no error
Wall-clock time	35 s	2 min 39 s

The wall-clock and temporal-check rows are recorded-run diagnostics; the state counts and no-error results were reproduced in a fresh TLC run.

E.3 nuSMV (Tier 1)

NuSMV 2.7.1 checks `DroneTier1.smv` on the same fixed-point lattice as TLC. The file contains six safety specifications (three `INVARSPEC`s and their three `LTLSPEC G` counterparts) plus the LTL recovery property; all seven are true.

The declared space is $65^3 \times 3^3 \times 2 \approx 1.48 \times 10^7$ states. Symbolic reachability reaches a fixed point at diameter 8 with $\approx 3.71 \times 10^6$ reachable states, matching TLC.

Table 9: nuSMV Tier 1 state space and BDD statistics.

Metric	**Value**
Declared state space	$2^{23.82} \approx 1.48 \times 10^7$
Reachable states	$2^{21.82} \approx 3.71 \times 10^6$
Reachability diameter	8
BDD nodes (initial set)	29
BDD nodes (transition relation)	861
BDD/ADD nodes after full property run	473 771

Table 10: nuSMV Tier 1 property results. A full-file run took 1.46 s in the fresh rerun; per-property timings were not separately measured.

Property	**Encoding**	**Result**
P1 Geofence safety	`INVARSPEC SafeRegion`	true
Domain containment	`INVARSPEC !overflow`	true
P2 Speed bound	`INVARSPEC SpeedBound`	true
P1 Geofence safety	`LTLSPEC G SafeRegion`	true
Domain containment	`LTLSPEC G !overflow`	true
P2 Speed bound	`LTLSPEC G SpeedBound`	true
P3 Recovery	`LTLSPEC` $\square(B \rightarrow \Diamond S_{\text{in}})$	true

The reachable count, diameter, property truth values, and BDD-cluster sizes were reproduced with `print_reachable_states` and `print_fsm_stats`.

E.4 nuSMV (Tier 2)

NuSMV checks `DroneTier2.smv` on the same declared state space as Tier 1. The file contains eight safety specifications (four `INVARSPEC`s and four `LTLSPEC G` formulas) plus recovery; all nine are true.

Symbolic reachability reaches a fixed point at diameter 8 with $\approx 2.95 \times 10^6$ reachable states, matching the exact TLC count 2,949,788.

Table 11: nuSMV Tier 2 state-space statistics.

Metric	**Value**
Declared state space	$2^{23.82} \approx 1.48 \times 10^7$
Reachable states	$2^{21.49} \approx 2.95 \times 10^6$
Reachability diameter	8
BDD nodes (initial set)	54
BDD nodes (transition relation)	1,415 + 256 (2 clusters)
BDD/ADD nodes after full property run	287 032

Table 12: nuSMV Tier 2 property results. A full-file run took 5.66 s in the fresh rerun; per-property timings were not separately measured.

Property	**Encoding**	**Result**
P1 Geofence safety $S \setminus O$	`INVARSPEC SafeRegion`	true
Domain containment	`INVARSPEC !overflow`	true
P2 Speed bound	`INVARSPEC SpeedBound`	true
P4 Obstacle avoidance	`INVARSPEC !InObstacle`	true
P1 Geofence safety $S \setminus O$	`LTLSPEC G SafeRegion`	true
Domain containment	`LTLSPEC G !overflow`	true
P2 Speed bound	`LTLSPEC G SpeedBound`	true
P4 Obstacle avoidance	`LTLSPEC G !InObstacle`	true
P3 Recovery	`LTLSPEC` $\square(B \rightarrow \Diamond S_{\text{in}})$	true

Here `SafeRegion` is the nonconvex predicate `Geofence` $\wedge \neg$`InObstacle`, so the Tier 1 geofence check (**P1**) and the new obstacle check (**P4**) are reported separately.

Remark 6 (Non-vacuity of the dilation). To confirm **P4** is not vacuously satisfied, we re-ran the model with the alignment predicate computed on the bare obstacle slab $[\hat{o}_j^{\min}, \hat{o}_j^{\max}]$ instead of the dilated slab $[\hat{o}_j^{\min} - \hat{\delta}_O, \hat{o}_j^{\max} + \hat{\delta}_O]$. NuSMV then returns **P1** and **P4** *false*, exhibiting the diagonal corner counterexample of Section C.1: from the admissible initial state $\hat{\mathbf{q}} = (18, 18, 32)$, $\hat{\mathbf{v}} = (-8, -8, 0)$ the drone coasts to $\hat{\mathbf{q}} = (7, 7, 32) \in \hat{O}$ before braking engages. The dilation is therefore load-bearing for the Tier 2 guarantee.

F Code Repositories

The TLA+, nuSMV, and Lean code can be found in the repository: `https://github.com/eyang07/droneV`

The SystemVerilog implementation, including the synthesizable RTL modules, safety monitor, testbench, and simulation script, is available in the project repository: `https://github.com/CanAfacan/CE356_backupstorage`.

References

Tudor Achim and Jonathan Thomm and 21 other authors. Aristotle: Imo-level automated theorem proving. *arXiv preprint arXiv:2510.01346*, 2025. doi: 10.48550/arXiv.2510.01346.

Dejanira Araiza-Illan and Kerstin Eder. Evaluating model checking approaches to verify stability of control systems in simulink, 2015.

Stephen Boyd. Ee263: Introduction to linear dynamical systems, lecture 13: Linear dynamical systems with inputs and outputs, n.d. URL `https://ee263.stanford.edu/lectures/linsys.pdf`. Lecture notes.

Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992. doi: 10.1016/0890-5401(92)90017-A.

Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986. doi: 10.1145/5397.5399.

Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, Cambridge, MA, 1999. ISBN 978-0-262-03270-4.

Edsger W. Dijkstra. Notes on structured programming. Technical Report EWD249, Technische Hogeschool Eindhoven, 1970.

Fauna Robotics, Diego Aldarondo, Ana Pervan, Daniel Corbalan, Dave Petrillo, Bolun Dai, Aadhithya Iyer, Nina Mortensen, Erik Pearson, Sridhar Pandian Arunachalam, Emma Reznick, David Weis, Jacob Davison, Samuel Patterson, Tess Carella, Michael Suguitan, David Ye, Oswaldo Ferro, Nilesh Suriyarachchi, Spencer Ling, Erik Su, Daniel Giebisch, Peter Traver, Sam Fonseca, Mack Mor, Rohan Singh, Sertac Guven, Kangni Liu, Yaswanth Kumar Orru, Ashiq Rahman Anwar Batcha, Shruthi Ravindranath, Silky Arora, Hugo Ponte, Dez Hernandez, Utsav Chaudhary, Zack Walker, Michael Kelberman, Ivan Veloz, Christina Santa Lucia, Kat Casale, Helen Han, Michael Gromis, Michael Mignatti, Jason Reisman, Kelleher Guerin, Dario Narvaez, Christopher Anderson, Anthony Moschella, Robert Cochran, and Josh Merel. Fauna sprout: A lightweight, approachable, developer-ready humanoid robot, 2026.

Sidharth Hariharan, Christopher Birkbeck, Seewoo Lee, Ho Kiu Gareth Ma, Bhavik Mehta, Auguste Poiroux, and Maryna Viazovska. A milestone in formalization: The sphere packing problem in dimension 8, 2026.

Gerard J. Holzmann. The model checker spin. *IEEE Transactions on Software Engineering*, 23(5): 279–295, 1997. doi: 10.1109/32.588521.

John Lygeros, Claire Tomlin, and Shankar Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3):349–370, 1999.

Franco Mazzanti and Alessio Ferrari. Ten diverse formal models for a cbtc automatic train supervision system, 2018.

Chase Norman and Jeremy Avigad. Canonical for automated theorem proving in lean. In *16th International Conference on Interactive Theorem Proving (ITP 2025)*, volume 352 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:20, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi: 10.4230/LIPIcs.ITP.2025.14. URL `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITP.2025.14`.

Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (FOCS 1977)*, pages 46–57. IEEE, 1977.

Ragunathan Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: The next computing revolution. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 731–736, New York, NY, USA, 2010. ACM. ISBN 9781450300025. doi: 10.1145/1837274.1837461.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.