Drone Neutralization Impact Prediction System

Technical Design Document

Mixed Emerging Technology Integration Lab (METIL)

Author: Ruba Ismail

Source Code: https://github.com/rubaismail/drone-decision-support-system

Demo Video: https://www.youtube.com/watch?v=SCa4DtVhcKs

1. Introduction

Counter–unmanned aerial system (C-UAS) technologies are increasingly deployed to mitigate risks posed by small unmanned aerial systems (sUAS) in civilian, industrial, and security-sensitive environments. While significant research and development has focused on detection, tracking, and neutralization, the **post-neutralization phase**, specifically the uncontrolled descent and impact of a disabled drone, remains comparatively underexplored.

Neutralization does not eliminate risk. Once propulsion is removed, a drone becomes a falling object whose impact location, timing, and severity depend on its physical state, environmental conditions, and the moment at which neutralization occurs. In urban or infrastructure-dense environments, this secondary risk can pose significant hazards to people and property.

This project addresses that gap by designing and implementing a **physics-based simulation system** that predicts post-neutralization descent behavior, estimates impact location and uncertainty, evaluates relative risk, and validates those predictions against actual simulated outcomes. The system is designed as a modular, extensible MVP, suitable for experimentation, training, and future research rather than immediate operational deployment.

2. Problem Motivation and Operational Gap

Existing C-UAS systems are typically optimized around three stages: detection, identification, and defeat. Publicly available analyses of counter-drone architectures

emphasize radar, RF sensing, and EO/IR tracking as primary enablers of these stages, but often treat neutralization as the terminal event in the engagement chain [1][2].

In practice, the moment of neutralization introduces a new class of risk. A drone disabled at altitude retains kinetic energy and horizontal momentum, and environmental factors such as wind can significantly alter its impact point. Despite this, few operational systems provide real-time tools for predicting where a neutralized drone will land, how uncertain that prediction is, or whether delaying neutralization could reduce risk.

This project was motivated by that identified gap. Rather than attempting to replicate existing detection or tracking capabilities, the focus was deliberately placed on **post-neutralization impact modeling**, an area where simulation-driven analysis can provide immediate insight and value.

3. Conceptual Approach and MVP Scope

3.1 Neutralization Assumptions

For the MVP, neutralization is modeled as **motor cutoff**, resulting in immediate loss of lift and uncontrolled descent. This assumption reflects outcomes common to several neutralization methods, including RF jamming and GNSS denial, while avoiding the complexity of modeling active control or thrust modulation.

While the codebase defines additional neutralization modes (e.g., partial thrust loss, explosive disablement, tethered capture) and associated parameter profiles, these are currently implemented as explicit placeholders and are not yet integrated into the active prediction logic.

3.2 Physics-Based Modeling over Machine Learning

Although machine learning is often proposed for trajectory prediction problems, it was intentionally excluded from the MVP. Effective ML-based prediction requires large, representative datasets with reliable ground truth. Such data is not readily available for post-neutralization drone falls, and prematurely applying ML would risk producing opaque or misleading results.

A deterministic physics-based approach was therefore chosen. This approach offers transparency, debuggability, and a clear path to validation. Importantly, the simulation itself

can later be used to generate labeled datasets for hybrid physics–ML approaches once sufficient data diversity exists.

4. Architectural Design and Separation of Concerns

4.1 Clean Architecture Principles Applied to Unity

Although Unity is not traditionally associated with Clean Architecture, the system was designed to follow its core principles: separation of concerns, dependency inversion, and scalability. Prediction logic is intentionally decoupled from Unity-specific components, allowing simulation to stand in for real-world sensor-derived state without entangling core logic.

The architecture is conceptually divided into four layers:

- **Core Logic**: Physics prediction, risk modeling, and decision support
- **Infrastructure**: Unity-specific providers for drone state, terrain height, and wind
- **Assembly & Control**: Orchestration of simulation flow, neutralization triggering, prediction execution, and simulation reset logic.
- **Presentation**: Visualization, UI, and camera control

This separation proved critical as the system evolved and complexity increased.

4.2 Provider-Based State Acquisition

In the simulation, Unity provides direct access to position and motion data. In real-world systems, however, drone state must be estimated from sensor fusion pipelines combining radar, EO/IR, and RF sensing [1][2][3].

To mirror this reality, the system relies on provider interfaces for drone state, ground height, and wind. Unity-based providers supply ground-truth values during simulation, but these can be replaced by sensor-driven implementations in future deployments without modifying prediction logic. This design choice aligns with operational C-UAS architectures, where sensing and reasoning are explicitly separated.

In real scenarios:

`Radar/EO/RF Sensor Fusion → Estimated State → DroneState`

In simulation:

`UnityDroneStateProvider → DroneState`

Because the prediction and risk pipeline consumes only the assembled `DroneState`, improvements in sensing modalities (e.g., better radar arrays, EO trackers, RF analyzers) can be integrated without modifying core prediction logic.

5. Core Data Representation

5.1 Drone State Modeling

The `DroneState` structure represents the authoritative physical state of the drone at the moment of prediction. It includes world position, velocity, mass, timestamp, and altitude above ground (AGL). A key design decision was treating AGL as a derived quantity rather than a raw coordinate.

Early in development, altitude was implicitly inferred from world coordinates, leading to systematic errors when real terrain was introduced. These issues closely mirror real-world challenges, where AGL must be computed by subtracting terrain elevation from an estimated absolute altitude using digital elevation models (DEMs) [4][5].

The final design resolves this by assembling drone state through a dedicated `DroneStateAssembler`, which queries terrain height and computes AGL consistently.

(more in section 12.1)

6. Physics-Based Fall Prediction Model

6.1 Predictor Inputs and Assumptions

The `FallPredictor` consumes a snapshot of the drone's physical state at the moment of neutralization, including:

- altitude above ground (AGL),
- full 3D velocity vector,
- Drone mass,
- environmental wind input.

A critical design decision was treating this state as **immutable during prediction**. The predictor does not query Unity objects, physics components, or transforms directly. Instead,

all required data is supplied through the `DroneState` structure. This enforces separation between prediction and simulation and guarantees that repeated predictions over the same state produce identical results.

Neutralization is assumed to cause immediate loss of lift, with no residual thrust, control authority, or active stabilization. This assumption establishes a clean physical baseline and avoids introducing hidden dependencies on drone-specific control logic.

6.2 Vertical Motion and Time-to-Impact

Vertical descent is modeled using constant-acceleration kinematics under gravity:

$$y(t) = y_0 + v_{0y}\, t - \frac{1}{2}\, g\, t^2$$

where y_0 is altitude above ground, v_{0y} is initial vertical velocity, and g is gravitational acceleration. Solving for y(t) = 0 yields time-to-impact. A closed-form solution was chosen to ensure numerical stability and deterministic behavior.

6.3 Horizontal Motion and Wind Effects

Horizontal displacement is computed independently in the XZ plane:

$$\vec{x}(t) = \vec{v}_0\, t + \frac{1}{2} + \vec{a}_{wind}\, t^2$$

In the MVP, wind is modeled as a **constant horizontal influence** derived from wind speed and direction rather than as a full aerodynamic force model. This simplification was chosen to balance realism with interpretability and to avoid introducing drag parameters that cannot yet be validated empirically.

6.4 Impact Point and Horizontal Uncertainty Estimation

The predicted impact point is computed by advancing the horizontal motion model to the predicted time-to-impact. Rather than treating this location as a precise point estimate, the system explicitly represents horizontal uncertainty as a drift radius.

In the MVP, this uncertainty radius is derived from the predicted horizontal motion under wind influence and serves as a conservative spatial envelope around the nominal impact point. It is intended to capture the practical reality that small deviations in wind conditions,

aerodynamic behavior, or state estimation can produce meaningful differences in impact location, even when the underlying prediction model is deterministic.

While this radius is not the result of probabilistic sampling or formal uncertainty propagation, it provides an interpretable and operationally useful approximation of spatial risk. The predicted drift radius is propagated directly into the visualization layer, where it defines the size of the ground-projected heatmap disk.

This design choice prioritizes clarity and usability at the MVP stage, while establishing a clear extension point for future probabilistic or data-driven uncertainty modeling.

6.4 Impact Velocity and Energy

At impact time, vertical and horizontal velocity components are reconstructed to compute total impact velocity. Impact energy is then calculated as:

$$E = \frac{1}{2} mv^2$$

This value serves as the primary physical proxy for impact severity in the MVP.

6.5 Role of the Predictor in the Overall System

The `FallPredictor` is invoked by the `PredictionRunner` at the moment of neutralization, and its outputs feed directly into risk classification and recommendation logic,impact uncertainty visualization, and validation/comparison tooling.

As such, it functions as the **single source of predicted truth** within the system. All architectural decisions surrounding state assembly, environmental modeling, and validation were ultimately driven by the need to support this predictor reliably.

7. Risk Modeling and Decision Support

7.1 Dual Risk Representation

Risk is represented both as a continuous normalized value (`risk01`) and as a qualitative category (Low, Medium, High). The continuous representation enables smooth visual scaling and computation, while the qualitative representation improves operator readability.

The mapping from predicted impact energy to both `risk01` and qualitative risk level is currently implemented using fixed heuristic thresholds (`EnergyLowJ = 100J`, `EnergyHighJ`

= 800J). These thresholds provide a consistent internal scale for relative risk comparison but are not empirically calibrated to real-world injury or damage models. As such, they are treated as MVP-level approximations, intended to support visualization, comparison, and architectural validation rather than authoritative risk assessment. Future work will replace these heuristics with data-driven or context-aware calibration.

7.2 Neutralization Timing Recommendations

Beyond producing a single static prediction, the system includes a recommendation mechanism that evaluates whether delaying neutralization could reduce downstream impact risk. This capability is implemented by the `ImpactRecommendationEngine` and builds directly on the nature of the physics-based prediction model.

Conceptually, the recommendation process treats neutralization timing as a controllable decision variable. The system evaluates a small set of alternative neutralization moments in the near future and compares their predicted outcomes against the baseline (immediate neutralization).

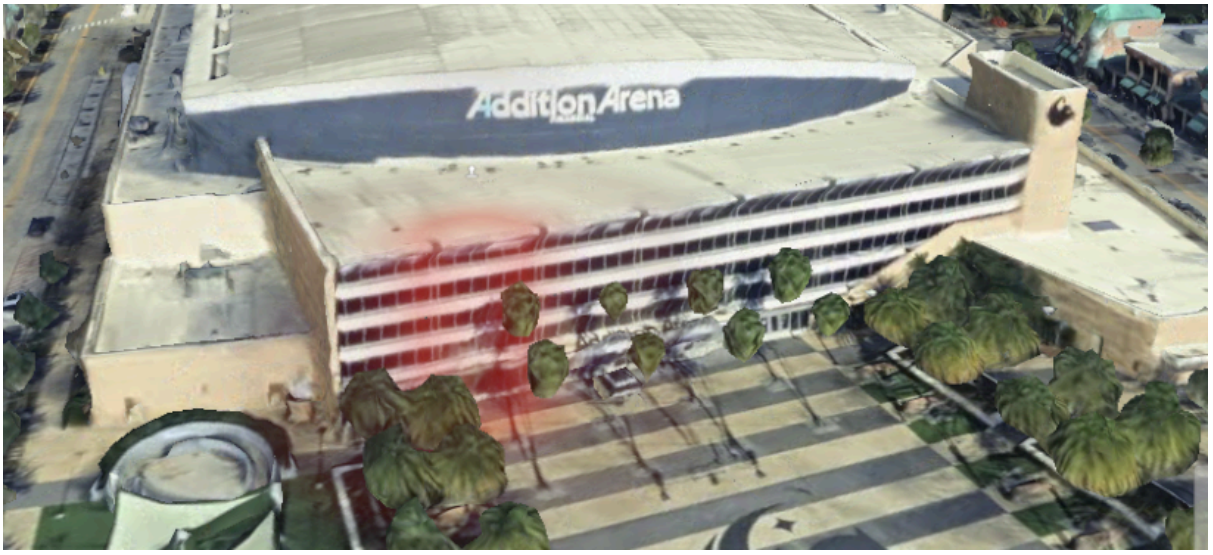
Importantly, this process does **not** attempt continuous optimization or trajectory planning. The system evaluates a discrete, bounded set of candidate delays using deterministic prediction rather than solving a global optimization problem. The recommendation engine evaluates short candidate delays by analytically advancing the drone’s vertical motion under gravity while preserving the current velocity state and environmental wind assumptions. No active control behavior or maneuvering is modeled during this delay

By framing neutralization timing as an evaluable parameter rather than a fixed event, the system demonstrates how even simple physics-based prediction can support higher-level decision reasoning. This capability establishes a foundation for future extensions, such as method-specific recommendations or context-aware timing constraints, without requiring changes to the core prediction model.

8. Visualization and User Interaction

8.1 Spatial Risk Visualization

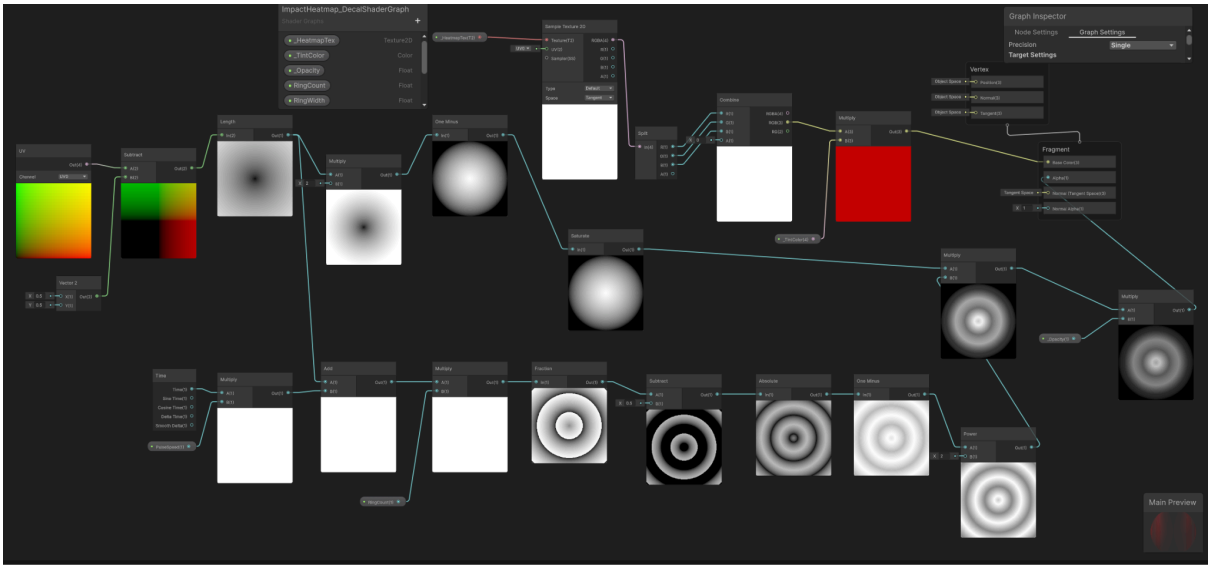
Impact uncertainty is visualized using a ground-projected heatmap disk. Rather than displaying a single impact point, the system communicates uncertainty spatially by mapping predicted horizontal drift to disk radius.



The visualization uses Unity's *URP DecalProjector* to conform to uneven terrain without modifying meshes. Color opacity encodes relative risk, enabling rapid interpretation without requiring numeric analysis.

8.2 Shader Design

The heatmap shader implements radial falloff and animated concentric rings to communicate uncertainty. Shader parameters are controlled through material property blocks to avoid instancing issues and ensure compatibility across rendering pipelines.



9. Validation and Outcome Comparison

A defining feature of the system is explicit, code-driven validation of predicted outcomes against actual simulated results.

9.1 Validation Trigger and Impact Detection

Actual impact is detected exclusively through physics collision events, ensuring that validation is grounded in the simulation's physical truth rather than inferred state.

The `ImpactOutcomeRecorder` script is responsible for:

- detecting the first ground collision following neutralization,
- recording the exact world-space impact position,
- capturing the simulation timestamp at impact.

This design ensures that validation is event-driven and not dependent on predicted timing or thresholds, eliminating confirmation bias in the evaluation process.

9.2 Measurement of Actual Impact Metrics

Upon impact detection, `ImpactOutcomeRecorder` computes the following actual outcome metrics:

- **Actual Time-to-Impact (TTI):**
  Measured as the elapsed time between neutralization and the collision event.
- **Actual Impact Velocity:**
  Derived from the drone's Rigidbody velocity at the moment of collision.
- **Actual Impact Energy:**
  Computed using the same kinetic energy formulation used during prediction.

9.3 Prediction vs Actual Comparison

The `ImpactComparisonData` structure is used to store paired predicted and actual values, including:

- predicted vs actual time-to-impact,
- predicted vs actual impact energy,
- horizontal error distance between predicted and actual impact points.

Horizontal error is computed in the ground plane (XZ), reflecting the most operationally relevant uncertainty for impact risk assessment.

This comparison data is populated immediately after impact and passed to the UI layer for visualization.

9.4 Visualization of Validation Results

The validation results are presented to the operator through the comparison UI, managed by the `ComparisonPanelController` and `UIStateController`.

Additionally, the `CameraDirector` transitions the camera from operator view to an impact-focused perspective.

9.5 Role of Validation in System Evolution

This validation loop played a central role in system refinement. Early validation runs exposed several non-obvious modeling and architectural issues, including:

- incorrect altitude-above-ground (AGL) computation due to implicit terrain assumptions,
- inconsistent ground snapping between prediction and visualization,
- double application of wind effects across simulation and prediction paths.

These issues were identified because predicted outcomes were directly compared to actual impact data, and resolving them required architectural changes such as the introduction of centralized state assembly.

10. Real-World Sensing and System Mapping

In real deployments, drone state is estimated rather than measured directly. Radar provides range and Doppler-based velocity estimates, EO/IR sensors provide angular tracking, and RF sensing may provide identification or bearing information when available [1][2][3]. These inputs are fused using tracking filters to produce a continuous state estimate over time [6].

The system's provider-based architecture mirrors this reality. Unity providers supply ground-truth state during simulation, but can be replaced with sensor-fusion-based providers without altering prediction logic. Terrain databases and meteorological data sources would replace simulation equivalents for real-world deployment [4][7].

11. Observed Recommendation Behavior and Model Limitations

In practice, the recommendation engine frequently produces a recommendation to neutralize immediately. This outcome arises directly from the ballistic descent assumptions used in the MVP, the short bounded delay window evaluated by the recommendation engine, and UI-level thresholding that suppresses negligible delay recommendations. When delayed neutralization does not produce a meaningful reduction in predicted impact energy, the system explicitly returns an immediate neutralization recommendation

Because the current model treats post-neutralization descent as ballistic motion under gravity and wind, delaying neutralization typically results in either increased altitude at the moment of neutralization, or increased horizontal displacement prior to descent, both of which tend to increase impact energy, drift radius, or overall spatial risk. Under these conditions, earlier neutralization often minimizes time-to-impact and limits horizontal drift, producing a lower or equivalent predicted risk compared to delayed alternatives. The recommendation engine reflects this reality by comparing predicted outcomes rather than forcing a non-trivial recommendation. When delayed neutralization does not yield a measurable improvement, the system explicitly returns an immediate neutralization recommendation.

This outcome also highlights an important limitation of the MVP: without modeling lift, glide behavior, partial thrust loss, or active control following neutralization, the system cannot capture scenarios in which delayed neutralization meaningfully reduces risk. These scenarios are expected to emerge only when higher-fidelity aerodynamics or neutralization-method–specific dynamics are introduced.

By surfacing immediate neutralization as the dominant recommendation under ballistic assumptions, the system provides a transparent and defensible baseline against which more advanced models can be evaluated in future work.

12. Engineering Challenges and Lessons Learned

The development of the MVP was highly iterative, with multiple rounds of refactoring driven by discrepancies between predicted and observed outcomes. Many of the most significant challenges were not apparent at the outset and only became visible once prediction results

were validated against actual simulated behavior. This section documents the major engineering challenges encountered and the lessons that informed the final system design.

12.1 Altitude Above Ground (AGL) Misinterpretation

One of the earliest and most impactful challenges involved correctly determining the drone's altitude above ground (AGL). Initial implementations implicitly assumed that the drone's vertical world coordinate could be interpreted directly as altitude. This assumption held in flat, synthetic environments but failed immediately once real terrain was introduced through Cesium integration.

In environments with non-zero terrain elevation, this led to systematic errors including:

- negative AGL values when the drone was near the ground,
- inflated predicted fall times,
- and incorrect impact energy estimates.

The core issue was a conceptual one: AGL is not a coordinate; it is a derived quantity. As mentioned earlier in the report, In real-world systems AGL is computed by subtracting terrain elevation from an estimated absolute altitude, and the same principle needed to be applied in simulation.

This realization led to the introduction of a dedicated `DroneStateAssembler`, which computes AGL by querying terrain height via a `GroundHeightProvider`.

Lesson learned: Never conflate world coordinates with physical meaning. Derived quantities must be computed explicitly and consistently.

12.2 Inconsistent Ground Snapping Between Prediction and Visualization

A second major issue arose from inconsistent ground references across system components. Early versions of the system snapped predicted impact points to ground height using one method, while visualization components independently queried terrain height for rendering purposes.

This resulted in subtle but confusing mismatches where the predicted impact disk appeared above or below the actual ground surface and validation error metrics appeared inflated due to mismatched reference frames.

The root cause was decentralized ground-height logic. Different components were querying terrain height independently, each with slightly different assumptions and sampling methods.

The resolution was to **c**entralize all ground-height queries behind a single provider interface and ensure that both prediction and visualization consumed the same snapped impact position. This enforced a single source of truth for ground reference throughout the system.

Lesson learned: Shared physical references must be centralized. Duplication of “simple” queries often leads to systemic inconsistencies.

Summary of Engineering Takeaways

Collectively, these challenges reinforced several core engineering principles:

- Physical meaning must be explicit, not inferred.
- Environmental effects require centralized ownership.
- Validation drives architecture, not the other way around.
- Prediction systems must be designed to surface error, not conceal it.

13. Future Directions and System Extensions

While the MVP establishes a validated baseline for post-neutralization impact prediction, the system was intentionally designed to support incremental expansion toward higher-fidelity modeling, broader operational relevance, and data-driven refinement. The following future directions build directly on the current architecture and address known limitations of the MVP without undermining its validated core.

13.1 Neutralization-Method–Aware Descent Modeling

The MVP models neutralization exclusively as motor cutoff, resulting in ballistic descent under gravity and wind. In practice, different neutralization methods produce distinct descent

behaviors. For example, partial propulsion loss may result in glide-like motion, while kinetic disablement may alter mass distribution or induce tumbling.

Future work will introduce **neutralization-method–specific descent models**, implemented by extending the existing `NeutralizationMethod` enum and introducing strategy-based prediction logic. Rather than modifying the core predictor directly, each neutralization mode can supply its own parameter set or motion model, preserving separation of concerns.

This approach allows the system to evaluate not only *whether* to neutralize, but *how* the choice of neutralization method influences downstream risk.

13.2 Higher-Fidelity Physics Modeling

The current prediction model intentionally omits aerodynamic drag and rotational dynamics to preserve interpretability and closed-form solvability. However, as validation data accumulates, higher-fidelity physics can be introduced incrementally.

Planned extensions include:

- quadratic drag modeling based on cross-sectional area and air density,
- tumbling and attitude-dependent drag coefficients.

These effects can be incorporated as optional numerical integration layers that operate alongside the existing closed-form solution.

Importantly, the validation pipeline already in place enables empirical evaluation of whether added complexity produces meaningful accuracy improvements.

13.3 Data-Driven and Hybrid Physics–ML Prediction

While machine learning is often proposed as a solution for trajectory and impact prediction problems, its effective application requires far more information than the deterministic physics variables used in the MVP. In particular, purely physics-based inputs such as position, velocity, mass, and wind are insufficient to capture the full range of variability observed in real-world post-neutralization drone behavior.

Machine learning models rely on high-dimensional input spaces to learn complex, nonlinear relationships. For post-neutralization descent modeling, this would require variables that are either unavailable or highly uncertain at the MVP stage, including:

- **Aerodynamic properties**, such as drag coefficients and cross-sectional area, which vary significantly across drone designs and orientations.
  Attitude and rotational state, including pitch, roll, yaw, and angular velocity, which strongly influence tumbling behavior and aerodynamic drag.
- **Structural asymmetries and damage states**, particularly for kinetic or partial neutralization methods.
- **Time-varying wind fields**, including gusts, shear, and turbulence, rather than a single constant wind vector.
- **Control system failure modes**, which affect how residual thrust or stabilization may influence descent.
- **Environmental interactions**, such as wake effects near buildings or terrain-induced airflow disturbances.

Without reliable access to these variables, an ML model would be forced to infer missing information indirectly, increasing the risk of overfitting, instability, and poor generalization. In such cases, machine learning may appear to perform well in narrow scenarios while failing unpredictably outside its training distribution.

For these reasons, the MVP deliberately prioritizes physics-based modeling, which provides interpretability, repeatability, and a clear baseline for validation. The physics model explicitly encodes known relationships and makes its assumptions visible, allowing systematic identification of modeling errors through comparison with observed outcomes.

A future hybrid approach is envisioned once sufficient validated data is available. In this paradigm, machine learning would not replace physics, but instead learn residual corrections to the physics-based predictions. For example, an ML model could be trained to predict systematic deviations between predicted and actual impact locations or times, conditioned on a richer set of input variables captured during simulation or real-world operation.

Crucially, the current system already supports this progression. The validation pipeline records predicted and actual outcomes in structured data (`ImpactComparisonData`), enabling the collection of labeled datasets required for supervised learning. As additional variables—such as orientation, drag parameters, and higher-resolution wind models—are incorporated into the simulation, they can be appended to the feature set without disrupting the existing prediction architecture.

This staged approach ensures that machine learning is introduced **only when it adds measurable value**, grounded in validated data rather than speculative modeling.

13.4 Environment-Aware Risk Modeling

The MVP's risk model is intentionally intrinsic, based solely on impact energy. In real-world deployments, risk is heavily influenced by environmental context, including population density, building type, and critical infrastructure proximity.

Future extensions will integrate geospatial data sources to weight impact risk based on:

- nearby structures and building footprints,
- population density estimates,
- infrastructure classification (e.g., roads, power systems).

This would transform risk assessment from a purely physical metric into a context-aware hazard model, enabling more informed decision-making in urban environments.

13.5 Sensor-Driven Deployment and Real-World Integration

While the current system uses Unity-based providers for state acquisition, the architecture is explicitly designed for integration with real sensor pipelines. Future work includes replacing simulation providers with sensor-driven implementations that ingest radar-based tracking data, EO/IR visual tracking outputs, RF detection and identification signals.

These inputs would be fused into an estimated `DroneState` using standard tracking filters, enabling the prediction and visualization pipeline to operate on live data. This progression would allow the system to transition from simulation-only evaluation to field-deployed experimentation.

13.6 Training, Planning, and Scenario Analysis

Beyond operational use, the system is well-suited for training and planning applications. Future development can enable scenario replay and comparison across multiple neutralization strategies, what-if analysis for timing and method selection,and structured datasets for after-action review.

13.7 Evaluation of Model Trust and Operator Interpretation

As predictive systems become more complex, understanding operator trust and interpretation becomes increasingly important. Future work may include evaluating how

different visualization strategies, uncertainty representations, and recommendation formats influence operator decision-making.

This human-centered extension ensures that increased model sophistication translates into improved operational outcomes rather than cognitive overload.

14. Conclusion

This project demonstrates that post-neutralization impact prediction is both feasible and operationally meaningful when approached through a physics-grounded, validation-driven design. By explicitly modeling post-neutralization descent, incorporating environmental effects such as wind, and validating predicted outcomes against observed simulation results, the system moves beyond speculative estimation and toward measurable, interpretable risk assessment. The use of spatial uncertainty visualization further enables intuitive understanding of impact risk, while the modular, provider-based architecture ensures scalability from simulation to real-world sensor integration. Collectively, the MVP establishes a strong foundation for future research, training applications, and extended operational exploration, particularly in environments where understanding downstream risk is as critical as the act of neutralization itself.

References

[1] J. A. Besada et al., “Review and Simulation of Counter-UAS Sensors,” *Sensors*, vol. 21, no. 21, 2021.

[2] G. Poppinga et al., *C-UAS Detection, Tracking and Identification Technology*, European Commission Joint Research Centre, 2020.

[3] S. D. Blunt et al., “Radar Detection of Small Unmanned Aircraft Systems,” *IEEE Radar Conference*, 2017.

[4] NOAA National Centers for Environmental Information, *Digital Elevation Models*, 2020.

[5] NASA EarthData, *Digital Elevation and Terrain Models (DEM)*, 2021.

[6] S. Thrun, W. Burgard, D. Fox, *Probabilistic Robotics*, MIT Press, 2005.

[7] NOAA, *Wind Measurement and Atmospheric Modeling Resources*, 2020.