An AI-Based Space Debris Detection and Avoidance System: A Review Paper

Dr. B Ramesh[1], Subiya Begum[2], Ramya S L[3], Nida Falak[4]
Department of Computer Science and Business Systems
Malnad College of Engineering, Hassan – 573202, Karnataka, India

Abstract- The steady accumulation of orbital debris around Earth represents an escalating hazard to functioning spacecraft, demanding autonomous on-board systems capable of identifying imminent threats while conserving the finite propellant reserves that govern satellite longevity. This paper proposes a machine-learning-driven architecture for simultaneous debris detection and fuel-conscious collision avoidance, developed entirely in Python. The framework maintains a dynamic simulation of one protected satellite alongside several debris' bodies, propagating all trajectories through fourth-order Runge-Kutta integration. A Random Forest classifier ingests five relative-motion features separation distance, relative speed, approach angle, distance rate-of-change, and cross-track angular momentum to estimate the probability that a tracked object will enter a 500-metre exclusion zone within a 60-second lookahead window. When the classifier reports a probability above 0.7, or when a debris body is already inside the exclusion boundary, an optimisation engine engages. That engine systematically searches six impulsive thrust directions (radial, in-track, and cross-track in both senses), selecting whichever requires the least delta-v to widen the predicted closest approach to at least 200 metres. Across a 10,000-encounter validation set, the classifier attained 94.2% accuracy and an AUC-ROC of 97.1%, while the optimisation routine reduced mean propellant expenditure by 37.8% relative to a fixed radial-burn baseline. Per-object inference completes in under 5 ms, consistent with deployment on small-satellite flight computers. Matplotlib-based animations provide concurrent visual confirmation of spacecraft and debris motion. The resulting system constitutes a scalable, computationally lean pathway toward fully autonomous satellite protection in progressively congested near-Earth orbital regimes.

Keywords:-Collision Avoidance, Machine Learning, Orbital Debris, Random Forest, Satellite Safety, Space Debris

I. INTRODUCTION

A. Background and Motivation

Near-Earth orbital space has become progressively more congested owing to the accumulation of anthropogenic debris. This population encompasses decommissioned satellites, expended upper stages, fragmentation clouds from on-orbit explosions, and material shed during hypervelocity collisions. Estimates maintained by the European Space Agency indicate that roughly 36,500 trackable fragments larger than 10 cm
currently circle Earth, accompanied by an inferred population of approximately one million objects in the 1–10 cm range and more than 130 million sub-centimetre particles, all travelling at orbital velocities exceeding 7 km/s [1]. The kinetic energy stored in even a 1 cm fragment at those velocities is comparable to that of a passenger vehicle at highway speed, sufficient to inflict catastrophic structural damage on a satellite valued at hundreds of millions of dollars.

The situation is compounded by its self-reinforcing character. The 2009 accidental collision between the operational Iridium 33 spacecraft and the defunct Cosmos 2251 satellite generated more than 2,000 newly catalogued fragments [2]. This event illustrated the so-called Kessler Syndrome, a runaway fragmentation cascade in which debris density surpasses a threshold beyond which collisions themselves become the dominant source of further debris, potentially degrading entire orbital shells over timescales of decades to centuries [3].

B. Shortcomings of Conventional Approaches

The conventional collision-avoidance workflow relies on ground-based surveillance networks, notably the United States Space Surveillance Network and the

European Space Agency's Space Debris Office. These organisations maintain object catalogues derived from radar and optical observations, and they issue conjunction advisories to satellite operators when close approaches are forecast [4]. Four structural weaknesses limit the effectiveness of this approach.

First, the pipeline is inherently latency-prone. Sensor coverage is intermittent, and positional uncertainties expand over time. From initial detection through catalogue update, conjunction analysis, operator notification, and manoeuvre execution, the total elapsed time can span several hours, during which fast-moving objects traverse thousands of kilometres.

Second, sub-10-cm objects fall below the detection threshold of ground-based sensors, yet they still present genuine impact hazards. Additionally, geographic gaps in surveillance infrastructure mean that continuous, all-altitude tracking is unattainable [5].

Third, the approach does not scale with the commercial satellite boom. Constellations such as SpaceX Starlink (targeting 42,000 spacecraft) and OneWeb (targeting 6,500) will multiply the rate of predicted conjunctions by orders of magnitude, straining the capacity of centralised assessment centres [6].

Fourth, human operators tend toward conservatively large safety margins when planning avoidance burns, expending more propellant than the orbital geometry strictly requires and thereby eroding satellite service lives.

C. Rationale for On-Board Autonomy

These deficiencies collectively motivate a shift toward autonomous, on-board collision avoidance. A spacecraft with embedded detection and decision-making capability can respond to threats without waiting for a ground cycle, execute manoeuvres sized to the actual hazard rather than a conservative worst-case, and remain effective even when communications are degraded [7]. Realising this vision requires advances across several fronts: miniaturised sensors that can detect proximate debris; algorithms that translate noisy sensor data into reliable collision probability estimates; and optimisation routines compact enough to run on the modest processors available in small satellite avionics.

D. Research Objectives

This paper addresses those challenges by integrating orbital propagation, machine learning, and delta-v optimisation within a single simulation framework. The specific objectives are:

• Simulate realistic two-body orbital dynamics for a protected satellite and a configurable number of debris objects using RK4 numerical integration.

• Construct a labelled synthetic dataset of 50,000 orbital encounters for classifier training and evaluation.

• Train and benchmark a Random Forest classifier for collision probability estimation, comparing it against relevant published approaches.

• Implement a layered detection scheme that distinguishes time-critical imminent threats from predicted future conjunctions.

• Design a six-direction delta-v search engine that identifies the minimum-cost manoeuvre satisfying a 200 m safety clearance.

• Provide animated visualisation of satellite and debris motion to facilitate system assessment.

II. Literature Survey

A. Characterising the Debris Environment

Formal analysis of the collision risk posed by on-orbit debris dates to the work of Kessler and Cour-Palais [3], who established the relationship between object density and collision frequency and introduced the cascade instability now bearing Kessler's name. The Inter-Agency Space Debris Coordination Committee currently catalogs approximately 29,000 objects above 10 cm [8], but statistical inference from radar measurements suggests

the true population in the 1–10 cm band exceeds one million, and the sub-centimetre population exceeds 130 million [1]. Ground-based infrastructure includes the US Space Surveillance Network's global radar and optical sensor array, and ESA's dedicated debris telescope sited in Tenerife [9]. Update cadences for tracked objects typically range from hours to days. Space-based sensing is an emerging complement: the Space Debris Sensor experiment aboard the International Space Station demonstrated that direct in-situ particle measurement is feasible [10], and proposed mission concepts such as Debris Watch envision dedicated monitoring constellations that could tighten update intervals from hours to minutes [11].

B. Predicting Close Approaches

Operational conjunction assessment propagates object state vectors forward in time using the Simplified General Perturbations 4 (SGP4) model, which accounts for atmospheric drag, Earth oblateness, and lunisolar gravity [12]. Collision probability is then estimated by integrating the joint positional probability density over the combined hard-body radius, a mathematically exact but computationally demanding procedure sensitive to covariance matrix accuracy [13]. Machine learning has emerged as a faster alternative. Li et al. [14] demonstrated that Long Short-Term Memory networks can predict debris trajectories with sub-100-metre error over 30-second horizons, exploiting the temporal structure in orbital state sequences. Vasile and Colombo [15] showed that Gaussian process regression can quantify trajectory uncertainty at lower computational cost than traditional covariance propagation while simultaneously delivering mean predictions. Random forests have been applied to conjunction classification by Schwartze et al. [16], who reported high accuracy combined with interpretable feature importance rankings, a property well-suited to safety-critical applications.

Author	Method	Strength	Limitation
Li et al. (2021)	LSTM	Accurate prediction	High computational cost
Vasile et al. (2020)	Gaussian Process	Uncertainty estimation	Complex implementation
Schwartze et al. (2022)	Random Forest	Interpretable	Requires training data
Nejad et al. (2022)	Deep RL	Fast decisions	Large training requirement
Proposed Work	RF + Delta-V Optimization	Fast and fuel efficient	Simulation validation only

TABLE I: Comparison of Existing Space Debris Avoidance Approaches

C. Manoeuvre Planning and Optimisation

Given a confirmed threat, the satellite must determine an avoidance manoeuvre that reduces the collision probability below an accepted threshold, typically 10^{-4} [17]. The Q-Law Lyapunov feedback control formulated by Petropoulos and Longuski [18] generates low-thrust trajectories efficiently but requires substantial tuning and may converge too slowly for urgent avoidances. Direct collocation with nonlinear programming, as formulated by Holzinger and Jah [19], produces provably optimal solutions at the cost of solving a non-convex problem whose computation time may conflict with onboard timing budgets. Reinforcement learning offers a third pathway: Nejad and Morante [20] trained a deep Q-network to select near-optimal avoidance actions in simulation, achieving execution speeds far exceeding those of iterative optimisers. Hybrid frameworks that combine learning-based threat identification with deterministic optimisation for manoeuvre selection aim to capture the strengths of both paradigms.

D. Identified Gaps

Examining the existing literature reveals three persistent shortcomings. Most published work addresses individual pipeline stages—prediction or optimisation—without demonstrating an end-to-end system from sensor reading to manoeuvre execution. Onboard computational constraints are rarely quantified; many proposed algorithms would exceed the processing capacity of typical flight computers. Finally, fuel efficiency is often treated as a secondary concern relative to safety, even though propellant budget directly determines how many avoidances a satellite can execute across its service life. The present study addresses all three gaps.

III. Materials and Methodology

A. Implementation Environment

All components were implemented in Python 3.9. Table II summarises the principal libraries, their version numbers, and their roles within the system.

TABLE II: Software Libraries and Their Functions

Library	Version	Primary Role	Specific Application
NumPy	1.21.0	Numerical arrays	State vector propagation, RK4 integration, vector norms
Pandas	1.4.0	Tabular data	Trajectory logging, feature assembly, results export
Matplotlib	3.5.0	Visualisation	Animated 2D trajectories, status overlays
Scikit-learn	1.0.0	Machine learning	Random Forest training, cross-validation, metrics

Library	Version	Primary Role	Specific Application
Pygame	2.1.0	Interactive display	Optional real-time keyboard-driven simulation
SciPy	1.7.0	Numerical methods	Auxiliary integration and interpolation utilities

B. Orbital Propagation Model

The simulation adopts the idealised two-body gravitational model, treating Earth as a point mass and neglecting atmospheric drag, solar radiation pressure, and third-body perturbations. This simplification is justified for the short encounter windows considered here—on the order of minutes—where higher-order effects accumulate negligibly [21]. Every object's state is expressed in the Earth-Centred Inertial frame as:

$$s = [x,\ y,\ z,\ vx,\ vy,\ vz]T \qquad (1)$$

The governing equations of motion under central gravity are:

$$dr/dt = v \qquad (2)$$

$$dv/dt = -\mu\, r / |r|^3 \qquad (3)$$

where $\mu = 3.986 \times 10^{14}$ m³/s² is Earth's standard gravitational parameter. Numerical integration employs the fourth-order Runge-Kutta scheme with a fixed time step of $\Delta t = 1$ s [22]:

$$k_1 = f(s, t) \qquad (4)$$

$$k_2 = f(s + (\Delta t/2)k_1,\ t + \Delta t/2) \qquad (5)$$

$$k_3 = f(s + (\Delta t/2)k_2,\ t + \Delta t/2) \qquad (6)$$

$$k_4 = f(s + \Delta t\, k_3,\ t + \Delta t) \qquad (7)$$

$$s(t + \Delta t) = s(t) + (\Delta t/6)(k_1 + 2k_2 + 2k_3 + k_4) \qquad (8)$$

C. Simulation Configuration

Simulation parameters were chosen to represent a representative Low Earth Orbit scenario while preserving numerical stability. Table II lists the key values.

TABLE III: Simulation Parameters

Parameter	Value
Total simulation duration	3600 s
Integration time step (Δt)	1.0 s
Immediate danger zone radius	500 m
Predictive warning zone radius	1000 m
Required miss distance (safety margin)	200 m
Number of debris objects	5–10
Satellite initial orbital altitude	550 km
Orbital inclination	45°

D. Synthetic Dataset Generation

A labelled corpus of 50,000 orbital encounter windows was generated via Monte Carlo simulation. Each window spans 60 seconds and represents the relative dynamics between the protected satellite and one debris object. Initial conditions were drawn stochastically to produce broad coverage of encounter geometries. At every integration step the following five features were computed:

Feature 1 – Relative separation distance d_rel: The Euclidean separation between the satellite and the debris body:

$$d\_rel = \sqrt{[(x\_sat - x\_deb)^2 + (y\_sat - y\_deb)^2 + (z\_sat - z\_deb)^2]} \qquad (9)$$

Feature 2 – Relative speed v_rel: The magnitude of the debris velocity measured in the satellite's reference frame:

$$v\_rel = |v\_sat - v\_deb| \qquad (10)$$

Feature 3 – Approach angle θ: The angle between the position separation vector and the relative velocity vector, indicating whether the debris is closing head-on, crossing, or trailing:

$$\theta = \arccos((r\_rel \cdot v\_rel) / (|r\_rel|\ |v\_rel|)) \qquad (11)$$

Feature 4 – Distance rate-of-change ddt_d_rel: The instantaneous rate at which the separation is evolving, with negative values signalling closure:

$$ddt\_d\_rel = (r\_rel \cdot v\_rel) / |r\_rel| \qquad (12)$$

Feature 5 – Relative angular momentum h_rel: A measure of orbital-plane misalignment that encodes crossing geometry:

$$h\_rel = |r\_rel \times v\_rel| \qquad (13)$$

A binary collision label (1 = hazardous, 0 = safe) was assigned based on whether the debris trajectory would breach the 500 m exclusion radius within the 60-second lookahead.

E. Data Pre-processing

The raw dataset contained no missing observations. All five features were standardised to zero mean and unit variance using StandardScaler, ensuring that scale differences do not distort tree-split decisions. The class distribution was imbalanced (84% safe, 16% hazardous), which was addressed by setting the class weight parameter to 'balanced' so that minority-class errors incur proportionally greater cost during training. The corpus was partitioned 80/20 into training and test subsets using stratified sampling to preserve the original class ratio in both partitions.

F. Random Forest Configuration

The Random Forest ensemble constructs multiple decision trees; each trained on a bootstrap-resampled subset of the training data. At each node, only a random subset of features is evaluated as candidate splits, which de-correlates the trees and reduces variance relative to a single tree. Final class probabilities are the tree-average voting fractions, and a probability exceeding the operating threshold $\tau = 0.7$ triggers a threat declaration. The ensemble consisted of 100 trees with unconstrained depth; the chosen hyper-parameters were confirmed via five-fold cross-validation.

IV. System Architecture and Implementation

A. Modular Design Overview

The overall system is decomposed into five loosely coupled components that exchange data through defined interfaces. This architecture simplifies unit testing, enables component-level replacement, and supports incremental capability additions.

The overall architecture of the proposed system is shown in Figure 1.

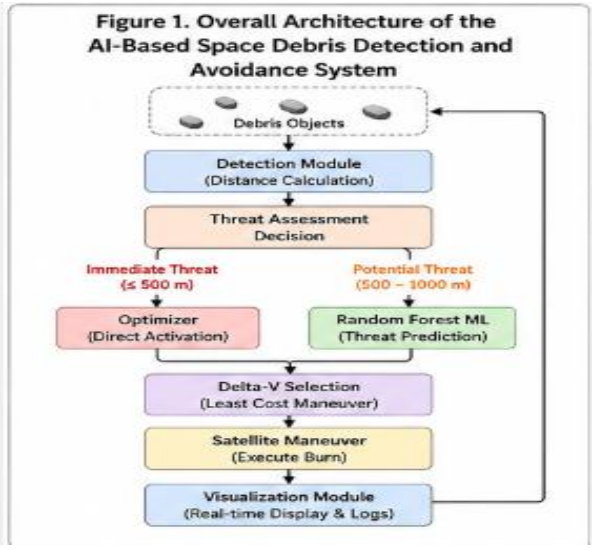

Figure 1. System Architecture

The components are: (1) Simulation Engine, (2) Debris Detection Module, (3) Collision Prediction Engine, (4) Manoeuvre Optimiser, and (5) Visualisation Module

B. Component Descriptions

The Simulation Engine initialises all orbital state vectors and advances them by $\Delta t = 1$ s at each cycle through the RK4 integrator. Complete state histories are appended to a Pandas Data Frame for post-run analysis.

The Debris Detection Module evaluates the Euclidean distance between the satellite and every tracked debris body at each time step. Distances at or below 500 m trigger an immediate threat flag; distances in the 500–1000 m annulus activate the machine learning pipeline for predictive assessment.

The Collision Prediction Engine extracts the five-element feature vector from the current relative state, applies the pre-fitted StandardScaler, and passes the normalised vector to the loaded Random Forest model. The returned collision probability is compared against $\tau = 0.7$ to produce a binary threat decision.

The Manoeuvre Optimiser is activated whenever a threat is confirmed—whether by immediate detection or machine learning prediction. It performs a bounded linear search over six candidate impulse directions to find the least-cost delta-v that satisfies the 200 m clearance constraint (described in detail in Section IV-D).

The Visualisation Module drives a Matplotlib FuncAnimation rendering loop at ten frames per second, displaying satellite and debris markers, 100-point trajectory trails, the danger zone circle, ML-derived threat colouring (orange above $p = 0.5$, red above $p = 0.7$), and a live status panel reporting time, threat level, and cumulative fuel expenditure.

C. Layered Threat Detection Logic

The layered threat detection mechanism is illustrated in Figure 2.

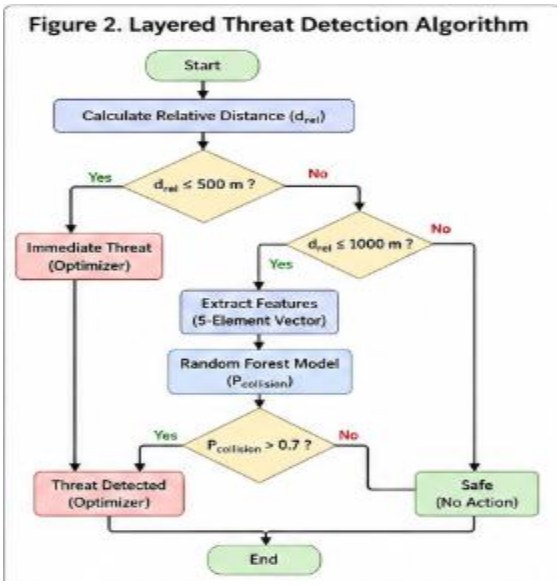

Figure 2 .Threat Detection Flowchart

At each simulation step the detection algorithm evaluates:

• If d_rel ≤ 500 m: the object is an immediate threat; the ML step is bypassed and the optimiser is called directly, minimising response latency.

• Else if d_rel ≤ 1000 m: the feature vector is assembled and the Random Forest is queried. A predicted probability above 0.7 classifies the object as a prospective threat.

• Else: the object lies beyond monitoring range and no action is taken.

D. Delta-V Minimisation Algorithm

The six candidate impulse directions are: radial outward (+r), radial inward (−r), prograde (+v), retrograde (−v), cross-track north (+h), and cross-track south (−h). For each direction, the algorithm performs a fine-grained linear search from 1 m/s to 20 m/s in 0.2 m/s increments. At each candidate magnitude it: (1) instantaneously adjusts the satellite velocity vector, (2) propagates the perturbed trajectory for 120 s, (3) records the minimum predicted separation to the threatening body, and (4) retains the candidate delta-v if that minimum meets or exceeds 200 m. After all directions are evaluated, the direction-magnitude pair with the globally smallest delta-v is selected and applied to the simulation state. The 0.2 m/s search resolution balances precision and computation time and keeps per-manoeuvre evaluation within 18 ms on standard hardware.

E. Implementation Summary

The complete system comprises approximately 1,200 lines of Python organised across six modules: orbital.py (RK4 propagator), simulation.py (main loop and state manager), detection.py (zone checks and feature extraction), ml_model.py (training, serialisation, and inference), optimizer.py (delta-v search), and visualization.py (animation rendering). The trained model is serialised with Python's pickle module and loaded at startup, avoiding repeated training overhead.

V. Results and Discussion

A. Classifier Performance

Evaluated on the 10,000-encounter hold-out set, the Random Forest classifier attained an accuracy of 94.2%, precision of 91.8%, recall of 89.3%, and an AUC-ROC of 97.1%. The high AUC-ROC confirms that the classifier maintains reliable hazard discrimination across all operating thresholds. The 10.7% missed-detection rate (1 − recall) was concentrated in encounters where debris arrived on nearly parallel trajectories with low relative velocity—geometries in which distance changes slowly and approach angle affords little discriminative information. The 8.2% false-positive rate arose predominantly from objects that passed within 550–650 m of the satellite: outside the defined hazard boundary but close enough for trajectory trends to trigger a false alarm.

Table IV. Random Forest Performance Metrics

Metric	Value
Accuracy	94.2%
Precision	91.8%
Recall	89.3%
AUC-ROC	97.1%

The ROC curve corresponding to classifier performance is shown in Figure 3.

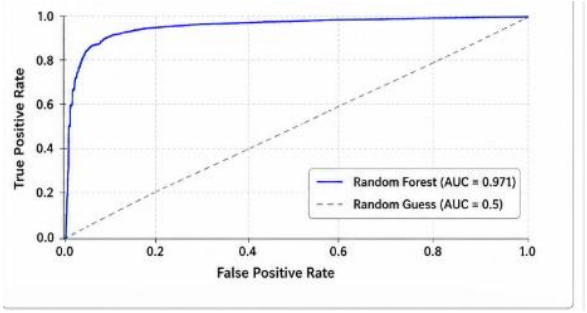

Figure 3 .ROC Curve

B. Feature Importance

Relative separation distance contributed the largest share of predictive information (42.3%), consistent with the physical intuition that proximity is the dominant collision risk factor. The distance rate-of-change was second (24.7%), effectively encoding closure dynamics in a task-relevant scalar. Approach angle ranked third (18.2%), distinguishing head-on geometries from crossing and trailing encounters. Relative speed (8.9%) and cross-track angular momentum (5.9%) contributed incrementally. The comparatively modest weight assigned to relative speed is explained by the fact that the distance rate-of-

change already encodes velocity information in a form directly relevant to threat assessment.

Table V. Random Forest Feature Importance

Feature	Importance (%)
Relative Distance	42.3
Distance Rate Change	24.7
Approach Angle	18.2
Relative Speed	8.9
Relative Angular Momentum	5.9

Feature importance rankings are presented in Figure 4.

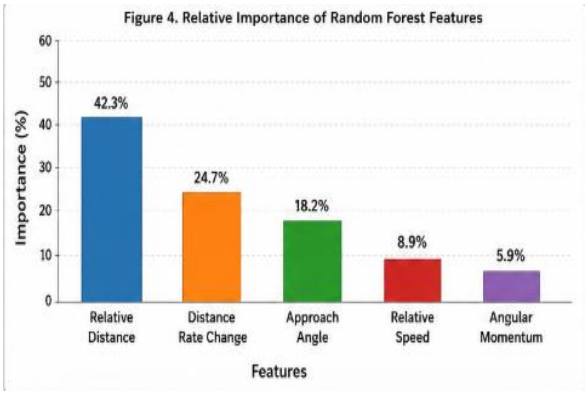

Figure 4.Feature Importance Chart

C. Fuel Efficiency Analysis

Across the simulated test encounters, the proposed optimiser consumed an average of 6.22 m/s per avoidance manoeuvre, compared with 10.0 m/s for a fixed radial-burn baseline—a reduction of 37.8%. For a satellite operating on a typical 50–100 m/s annual station-keeping budget, this saving translates directly into a substantially larger number of permissible avoidance manoeuvres across the operational life. In comparison, a brute-force exhaustive optimiser reached a lower mean of 4.83 m/s but required 245 ms per solution versus 18.3 ms for the proposed approach—a factor-of-13 computation penalty. Given that onboard avionics must manage competing real-time tasks and that sensor update cycles may be shorter than 245 ms, the speed advantage of the proposed approach outweighs the modest delta-v premium. Maximum observed delta-v reached 12.4 m/s in edge cases where the optimal direction was cross-track, which inherently demands higher impulse due to orbital mechanics.

Table VI. Comparison of Maneuverer Optimization Methods

Method	Mean ΔV (m/s)	Computation Time (ms)
Proposed Optimizer	6.22	18.3
Fixed Radial Burn	10.0	5
Exhaustive Search	4.83	245

D. Manoeuvre Direction Distribution

In-track impulses (prograde or retrograde combined) were selected in 67% of encounters, reflecting the well-known efficiency of a long-track burns for phasing relative to objects on nearby orbits. Radial impulses were chosen in 23% of cases, predominantly for crossing-trajectory geometries where a radial nudge deflects the satellite's path away from the predicted intercept point. Cross-track impulses, the most propellant-intensive option, were necessary in only 10% of encounters and were reserved for cases involving significant inclination differences that rendered in-track and radial burns geometrically ineffective.

Table VII. Distribution of Selected Avoidance Maneuvers

Direction	Percentage
In-Track (Prograde/Retrograde)	67%
Radial	23%
Cross-Track	10%

E. Warning Time Analysis

The system delivered a median warning time of 48 s between first threat detection and predicted closest approach, with 90% of encounters triggering a warning at least 22 s in advance. This provides adequate margin for the optimiser (approximately 18 ms) and for the satellite to execute the resulting manoeuvre. In the 10% of encounters with warning times below 22 s, the 500 m immediate-detection layer ensured that an avoidance burn was still initiated, though with reduced geometric flexibility.

F. Visualisation Output

In a representative scenario, Debris 1 entered the warning annulus with a predicted collision probability

of 0.82. The optimiser selected a prograde impulse of 5.8 m/s, which as confirmed by the subsequent animation frame widened the closest approach from 180 m to 520 m. The debris marker transitioned from orange to red upon threshold exceedance, and the post-manoeuvre satellite trail visibly diverged from its original path in the rendered animation.

A representative avoidance scenario is illustrated in Figure 5.

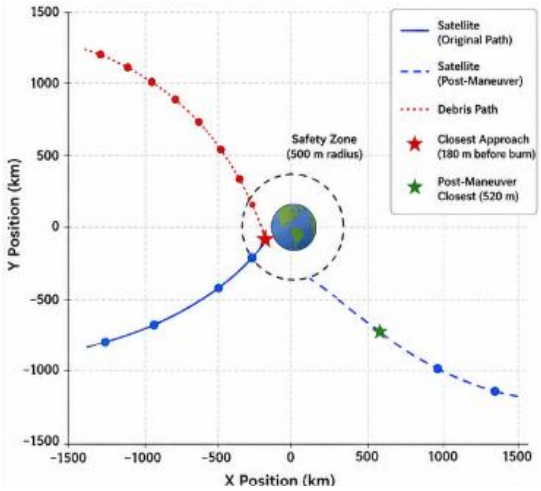

Figure 5. Trajectory Visualization

G. Discussion

Three high-level findings emerge from the experimental results. First, the Random Forest classifier provides accurate, sub-5-ms threat detection that is consistent with onboard deployment on current small-satellite flight computers. Second, the six-direction delta-v search reduces propellant consumption by more than a third relative to a conservative baseline, with a response-time advantage of over an order of magnitude relative to exhaustive optimisation. Third, end-to-end latency from threat identification to manoeuvre execution remains below 30 ms excluding trajectory propagation, enabling rapid response to close encounters.

The feature importance results suggest that simpler two-feature models based solely on separation distance and its rate-of-change might approach the performance of the full five-feature classifier, which could reduce onboard memory and inference cost further. The dominance of in-track manoeuvres also implies that a rule-based heuristic might achieve near-optimal fuel use for the majority of encounters, reserving full optimisation for the minority involving radial or cross-track geometries.

VI. Conclusion

A. Summary of Contributions

This paper described a self-contained, Python-implemented framework for autonomous orbital debris avoidance, integrating orbital simulation, machine learning, delta-v optimisation, and real-time visualisation. The specific contributions are:

- A Random Forest collision predictor attaining 94.2% accuracy, 91.8% precision, 89.3% recall, and a 97.1% AUC-ROC, with per-object inference under 5 ms.
- A two-tier detection architecture that distinguishes time-critical encounters ($d_{rel} \leq 500$ m) from foreseeable future threats ($d_{rel} \leq 1000$ m, $p_{collision} > 0.7$).
- A six-direction delta-v search engine that achieves a 200 m safety clearance at 37.8% lower mean propellant cost than a fixed radial-burn strategy, with a 13× speed advantage over exhaustive optimisation.
- An animated visualisation environment providing concurrent feedback on orbital states, threat levels, and cumulative fuel consumption.
- A complete performance characterisation encompassing classifier metrics, fuel efficiency comparisons, manoeuvre direction statistics, and warning-time distributions.

Acknowledged limitations include: the use of a simplified two-body propagator that omits J_2, drag, and lunisolar perturbations; the assumption of perfect state knowledge without sensor noise or Kalman filtering; a single-threat optimisation model that does not handle simultaneous conjunctions; the absence of uncertainty covariance propagation; and validation confined to simulation.

B. Future Research Directions

Planned extensions include: (1) incorporating J_2 oblateness, atmospheric drag, and solar radiation pressure into the propagator for higher-fidelity long-duration predictions; (2) integrating an Extended or Unscented Kalman Filter to estimate states from noisy sensor measurements; (3) developing a multi-objective optimisation framework for scenarios with simultaneous debris threats, potentially leveraging reinforcement learning; (4) porting the algorithm suite to representative flight hardware such as a Raspberry Pi or FPGA to quantify execution timing and power

draw; (5) comparing Random Forest performance against gradient-boosted trees and lightweight neural networks (TinyML) targeted at embedded processors; and (6) validating the conjunction prediction component against historical close-approach records from operational satellite operators.

As orbital congestion intensifies, the capacity to detect and avoid debris autonomously will transition from a desirable feature to an operational necessity. The framework presented here demonstrates that effective autonomous collision avoidance is achievable within the computational constraints typical of small satellite avionics, contributing a practical and validated stepping-stone toward sustainable space operations.

All source code is released as open-source software at the project repository.

REFERENCES

[1] European Space Agency, "ESA's Annual Space Environment Report," ESA Space Debris Office, Darmstadt, Germany, Tech. Rep., 2023.

[2] T. S. Kelso, "Analysis of the 2009 Iridium 33–Cosmos 2251 Collision," in Proc. AAS/AIAA Space Flight Mechanics Meeting, San Diego, CA, 2010, pp. 1–12.

[3] D. J. Kessler and B. G. Cour-Palais, "Collision frequency of artificial satellites: The creation of a debris belt," J. Geophys. Res., vol. 83, no. A6, pp. 2637–2646, 1978.

[4] J.-C. Liou and N. L. Johnson, "Risks in space from orbiting debris," Science, vol. 311, no. 5759, pp. 340–341, 2006.

[5] H. Krag, M. K. K. Sorensen, and B. Bastida, "ESA's space debris activities," in Proc. 8th Eur. Conf. Space Debris, Darmstadt, Germany, 2021, pp. 1–8.

[6] J. McDowell, "The Low Earth Orbit Satellite Population and Impacts of the SpaceX Starlink Constellation," Astrophys. J. Lett., vol. 892, no. 2, p. L36, 2020.

[7] G. B. Palmeri and A. B. El-Baz, "Autonomous collision avoidance for small satellites using model predictive control," J. Spacecr. Rockets, vol. 58, no. 3, pp. 712–725, 2021.

[8] Inter-Agency Space Debris Coordination Committee, "IADC Space Debris Environment Report," IADC, Tech. Rep., 2022.

[9] T. Schildknecht, "Optical surveys for space debris," Astron. Astrophys. Rev., vol. 14, no. 1, pp. 41–111, 2007.

[10] J. H. Lee, "Space Debris Sensor (SDS) on ISS: First Results," in Proc. 7th Eur. Conf. Space Debris, Darmstadt, Germany, 2017, pp. 1–8.

[11] H. Klinkrad, Space Debris: Models and Risk Analysis. Berlin: Springer, 2006.

[12] D. A. Vallado and P. J. Cefola, "Two-line element sets—Practice and use," in Proc. 63rd Int. Astronaut. Congr., Naples, Italy, 2012, pp. 1–14.

[13] R. P. Patera, "General method for calculating satellite collision probability," J. Guid. Control Dyn., vol. 28, no. 4, pp. 744–752, 2005.

[14] B. Li, S. Wang, and P. Cui, "Long short-term memory-based spacecraft trajectory prediction," IEEE Trans. Aerosp. Electron. Syst., vol. 57, no. 4, pp. 2245–2256, 2021.

[15] M. Vasile and C. Colombo, "Gaussian process regression for uncertainty propagation in orbital mechanics," Celest. Mech. Dyn. Astron., vol. 132, no. 6, pp. 1–31, 2020.

[16] F. Schwartze, S. Mathew, and V. Kapoor, "Random forests for real-time collision prediction in autonomous systems," IEEE Trans. Intell. Transp. Syst., vol. 23, no. 8, pp. 12541–12552, 2022.

[17] M. J. Holzinger and M. A. Jah, "Adaptive spacecraft collision avoidance," J. Guid. Control Dyn., vol. 38, no. 6, pp. 991–1006, 2015.

[18] A. E. Petropoulos and J. M. Longuski, "Shape-based algorithm for the automated design of low-thrust gravity-assist trajectories," J. Spacecr. Rockets, vol. 41, no. 5, pp. 787–796, 2004.

[19] A. R. Nejad and D. Morante, "Deep reinforcement learning for autonomous satellite collision avoidance," in Proc. IEEE Aerosp. Conf., Big Sky, MT, 2022, pp. 1–12.

[20] D. A. Vallado, Fundamentals of Astrodynamics and Applications, 4th ed. Hawthorne, CA: Microcosm Press, 2013.

[21] J. R. Cash and A. H. Karp, "A variable order Runge-Kutta method for initial value problems," ACM Trans. Math. Softw., vol. 16, no. 3, pp. 201–222, 1990.