AI Space Trajectory Optimizer

Mrs. N.S.Gite, Shubham Sarode, Shrawani Shewale, Apeksha Sonawane

Lecturer of Dept. Information Technology, K. K. Wagh Polytechnic, Nashik, India

Diploma Students Dept. of Information Technology, K. K. Wagh Polytechnic, Nashik, India

---***---

Abstract:

The rapid growth of satellites and debris objects in Low Earth Orbit (LEO) has made safe and responsive trajectory planning a critical requirement for modern space missions. Past collision events, such as the 2009 Iridium–Cosmos incident, have demonstrated how undetected trajectory conflicts can generate long-lasting debris and increase future mission risks. This project presents the AI Space Trajectory Optimizer, a hybrid trajectory planning system that combines a physics-based orbital mechanics engine with a lightweight machine learning warm-start predictor to generate collision-aware and fuel-conscious orbital transfer and insertion paths. The system accepts mission parameters and orbital elements as inputs and evaluates trajectory feasibility using numerical propagation, analytical collision checks, and delta-V estimation based on Tsiolkovsky's rocket equation. A trained AI model proposes an initial maneuver estimate, which is then refined and verified using the physics module to ensure constraint satisfaction and safety margins. Simulation tests with varied mission inputs and synthetic debris conditions demonstrated that the hybrid pipeline consistently produces valid maneuver plans while maintaining the transparency and auditability of calculations. Unlike purely iterative optimization approaches, this hybrid AI–physics method is designed to improve the planning responsiveness while preserving deterministic verification. The implementation includes a REST API backend, persistent mission logging, and a Three.js 3D visualization interface for the trajectory and debris inspection. The current limitations include dependence on simulated debris fields and simplified propulsion assumptions; however, the framework is extensible to real catalog data and higher-fidelity force models. This project demonstrates a practical, demonstrator-grade approach toward faster and safer autonomous trajectory planning for future space operations.

Keywords

trajectory optimization, orbital mechanics, space debris avoidance, hybrid AI–physics planning, delta-V estimation, collision risk analysis, autonomous mission planning, LEO safety

1. INTRODUCTION

If you look up at the night sky and imagine everything orbiting above you right now — active satellites, retired spacecraft, rocket bodies, paint chips, nuts, and bolts — the picture is considerably more cluttered than most people realize. The Low Earth Orbit (LEO), a band of space at an altitude of approximately 200–2000 km, has become particularly congested over the past decade. Commercial operators such as SpaceX, OneWeb, and Amazon are deploying constellations with hundreds or even thousands of satellites, and every launch adds to a growing population of objects that share the same narrow region of space.

This danger is not merely theoretical. In 2009, the operational Iridium 33 satellite and the defunct Cosmos 2251 collided unexpectedly over Siberia, producing a debris cloud that tracked thousands of objects. This event changed how the space industry thought about conjunction assessment and avoidance planning. What was once considered an unlikely edge case has become a concrete, documented risk [2]. Scientists had warned about this kind of cascading debris scenario decades earlier — the idea that each collision generates more fragments, which increases the chance of more collisions, and so on, until an entire orbital altitude becomes unusable. This is sometimes called the Kessler effect [1], and while it remains a long-term concern rather than an immediate crisis, it motivates the need for careful and proactive trajectory planning today.

The challenge for mission planners is that computing a good trajectory — one that reaches the target orbit, uses as little fuel as possible, and stays clear of everything else up there — is computationally expensive when done rigorously. Traditional physics-based solvers iterate through orbital equations until they converge on a valid solution, but this can take a while, especially when the debris environment is dense and many proximity checks are required at each time step. However, using a pure machine learning approach to plan trajectories is attractive from a speed standpoint, but it comes with a serious problem: there is no guarantee that the output satisfies the laws of orbital mechanics. A neural network might suggest a maneuver that is physically impossible or unsafe, and in space, these mistakes are not recoverable.

Our approach attempts to obtain the best of both worlds. We used a trained machine learning model to make a fast initial estimate of the required maneuver, which provided the physics solver with a much better starting point than a random guess. The physics engine then takes over, refines the solution, checks it against the debris environment, and confirms that it is physically valid. The result is a system that is meaningfully faster than running physics alone but is equally safe. The remainder of this paper discusses how we built it, what we found, and where we think it could go next.

2. LITERATURE SURVEY:

Most of what we know about planning orbital maneuvers traces back to the analytical solutions developed in the mid-twentieth century. The Hohmann transfer, for instance, provides the minimum-energy path between two circular orbits using only two propulsive burns. It is elegant and exact, but only works when the orbits are coplanar and nearly circular. Lambert's problem extends this concept to arbitrary endpoint-to-endpoint transfers with a fixed time of flight and forms the backbone of many trajectory optimization tools used today [6]. Vallado's textbook [1] remains one of the most referenced sources for the underlying astrodynamics — coordinate systems, orbit propagation, conjunction geometry — and we leaned on it heavily while building the physics validation side of our system.

The debris problem has been studied since the late 1970s. Kessler and Cour-Palais [2] published the original analysis showing that at some point, the density of orbiting objects could become self-sustaining — collisions producing fragments that cause more collisions — without any new launches at all. NASA's orbital debris office has continued this line of research, publishing quarterly reports that track the current debris population and assess the risk trends [5]. Reading some of these reports during our background research made it clear how actively this problem is being monitored at the institutional level.

Recently, researchers have begun experimenting with machine learning for spacecraft-related problems. Sharma et al. [7] explored how neural networks could approximate trajectory solutions faster than iterative solvers, showing promising speed improvements on certain transfer problem classes. The limitation acknowledged in this and similar studies is that the learned models generalize unpredictably outside their training distributions and offer no correctness guarantees. Wertz and Larson [4] frame mission design as an optimization problem across competing constraints — cost, schedule, performance, and risk — which is a useful conceptual lens for understanding why purely computational or purely data-driven approaches each fall short on their own. In reviewing the literature, we noticed that almost no one had tried to formally combine a trained predictor with a physics gate that must pass before the output is accepted. This gap is addressed in this study.

3. PROBLEM DEFINITION:

In other words, the problem we are trying to solve is as follows: given a spacecraft at some known initial orbit, a target orbit it needs to reach, a set of propulsion parameters (mass and specific impulse), and a surrounding debris environment, how do we compute the best transfer maneuver quickly and safely? Here, best refers to the minimum fuel use. “Quickly” means within a couple of seconds on the standard hardware. “Safely” means that no close approaches with tracked debris below a defined threshold distance are predicted.

The tension between speed and rigor is the core difficulty in this regard. A full numerical simulation is accurate but slow to perform. Machine learning models are fast but are not guaranteed to be accurate or physically valid. Standard iterative solvers, when started from a poor initial guess, can take many more iterations to converge than are necessary. The specific problem we designed around was whether a learned predictor can reduce the number of iterations required by the physics solver without ever allowing an unsafe or physically impossible trajectory to be returned to the user. Based on our experiments, the answer appears to be yes, at least for the class of LEO transfer scenarios we tested.

4. EXISTING SYSTEM:

The two most commonly referenced professional tools for mission trajectory planning are NASA's General Mission Analysis Tool (GMAT) and Analytical Graphics' Systems Tool Kit (STK). Both are powerful and widely used in the industry. The GMAT is open-source and capable of high-fidelity simulations that account for atmospheric drag, solar radiation pressure, third-body gravitational effects, and a range of numerical integrators. STK offers similar capabilities, with an emphasis on visualization and communications analysis. These are serious and well-validated tools.

However, they were not designed with lightweight, API-accessible, and quickly deployable use cases in mind. They run as desktop applications, require significant configuration to set up a new mission scenario, and are not easily integrated into a broader software pipeline or exposed as a web service. For a research prototype or student-level system, they are somewhat overkill, and the learning curve is steep. There is also no AI-assisted convergence acceleration in either platform ; every scenario starts the solver from scratch. For our purposes, we needed something leaner: a system that could accept a mission description via HTTP, return a validated trajectory plan within seconds, and display the result in a browser without requiring the installation of specialized software. None of the existing tools surveyed covered all of these requirements.

5. PROPOSED SYSTEM:

The system we built has three main components that work together in sequence. First, there is the AI predictor, a Random Forest regression model that takes the key mission parameters as input and produces a first estimate of the required delta-V. Second, there is a physics validation engine, which takes that estimate as its starting point, propagates the orbit numerically, runs debris proximity checks at each time step, computes the actual propellant cost using the Tsiolkovsky equation, and confirms that the plan is feasible. Third, the frontend is a Three.js web application that draws the Earth, planned orbit path, and debris field in an interactive 3D view that anyone with a browser can use.

The inputs accepted by the system are as follows: initial orbital altitude and inclination, target altitude and inclination, spacecraft wet mass (kg), specific impulse (s), whether atmospheric drag should be factored in, and whether propellant needs to be reserved for first-stage recovery. These are submitted as a JSON body to the POST endpoint on the Flask backend. The AI model processes a normalized version of the key numerical features — altitude difference, inclination change, mass, and Isp — and returns a delta-V estimate in meters per second. The physics engine then refines this through an iterative propagation loop, adjusting for drag if it is enabled and flagging any debris conjunctions it detects. The final output was a JSON response containing the validated delta-V, propellant mass required, collision clearance status, and mission feasibility flag. This response is also written to a SQLite database, so every mission attempt is logged and reviewable.

One design choice worth explaining is why we kept the AI model as a warm start rather than using it as the final answer. This is because a regression model, regardless of how well trained, can produce outputs that violate physical constraints in subtle ways, such as predicting a delta-V that is technically achievable but routes the spacecraft dangerously close to a debris cluster. By always running the physics gate after the AI prediction, we ensure that no unsafe results leave the system. AI speeds up processes, while physics maintains accuracy.

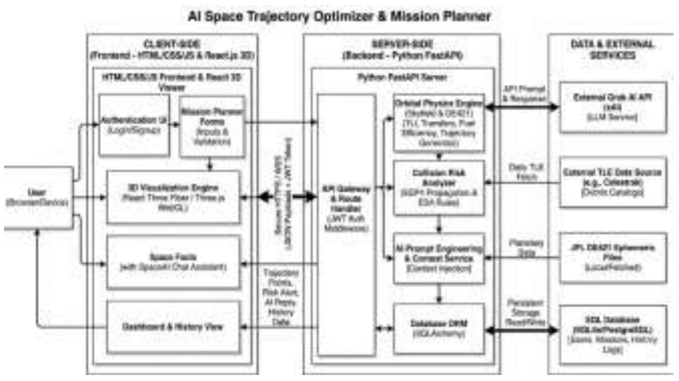
6. METHODOLOGY:

We began by synthetically generating a training dataset. We simulated 10,000 mission scenarios by sampling orbital parameters uniformly across realistic LEO ranges: initial altitudes between 200 and 1200 km, target altitudes in the same range, inclinations from 0 °to 98 °(covering sun-synchronous orbits), spacecraft masses from 500 to 5000 kg, and specific impulse values from 250 to 450 s, which span the range from simple chemical thrusters to more efficient bipropellant systems. For each scenario, we computed the ground truth delta-V using the analytical Hohmann transfer equation combined with the plane-change cost formula. These values were used as regression targets.

Once the model prediction is obtained, the physics engine takes over. It propagates the satellite's position numerically using two-body Keplerian dynamics over a configurable time window, stepping forward in 60-second intervals by default. At each step, the Euclidean distance between the satellite's predicted position and each debris object in the synthetic catalog is computed. If any distance falls below 10 km, which

is our chosen conjunction threshold, the system marks a collision risk and applies a lateral correction to the planned trajectory. The Tsiolkovsky rocket equation ($\Delta v = I_{sp} \times g_0 \times \ln(m_0 / m_1)$) then translates the validated delta-V into a propellant mass. When drag modeling was active, a velocity correction term proportional to the atmospheric density at the target altitude was added. When first-stage recovery is requested, approximately 18% of the total propellant mass is reserved for the return burn, reducing the available delta-V budget accordingly.

7. SYSTEM DESIGN:


AI Space Trajectory Optimizer & Mission Planner
CLIENT-SIDE (Frontend - HTML/CSS/JS & React.js 3D)
HTML/CSS/JS Frontend & React 3D Viewer
Authentication UI (Login/Signup)
Mission Planner Forms (Inputs & Validation)
3D Visualization Engine
Space Facts
Dashboard & History View
User (Browser/Device)
SERVER-SIDE (Backend - Python FastAPI)
Python FastAPI Server
API Gateway & Route Handler (JWT Auth Middleware)
Orbital Physics Engine
Collision Risk Analyzer
AI Prompt Engineering & Context Service (Context Injection)
Database ORM (SQLAlchemy)
Trajectory Points, Risk Alert, AI Reply, History Data
DATA & EXTERNAL SERVICES
API Prompt & Response
Daily TLE Fetch
Planetary Data
Persistent Storage Read/Write
External TLE Data Source (e.g., Celestrak) (Debris Catalog)
JPL DE421 Ephemeris Files (Local/Fetched)
SQL Database

The backend is a Python Fast API that exposes three primary endpoints. The main one, /plan_trajectory, accepts a POST request with the mission parameters and returns the full trajectory plan. The second endpoint, /mission_log, returns a list of all previously computed missions from the SQLite database, which is useful for reviewing how different configurations affect the outcome. Third, /debris_status returns the current synthetic debris catalog, which the frontend uses to draw the debris field in the 3D view. The AI model and physics engine both run inside the same Flask process; therefore, there is no inter-service communication overhead.

On the frontend, Three.js renders everything in a WebGL canvas. The Earth appears as a textured sphere using publicly available NASA imagery. The planned trajectory is drawn as a parametric curve in 3D space, colored green for safe segments, and red where a debris conjunction is detected and corrected. The debris objects appear as small yellow spheres distributed throughout the orbital shell. Users can click and drag to rotate the view, scroll to zoom, and hover over debris objects to view their catalog identifiers and estimated positions. All mission results, delta-V, propellant cost, and collision status, appear in a side panel that updates in real time when a new trajectory is submitted.

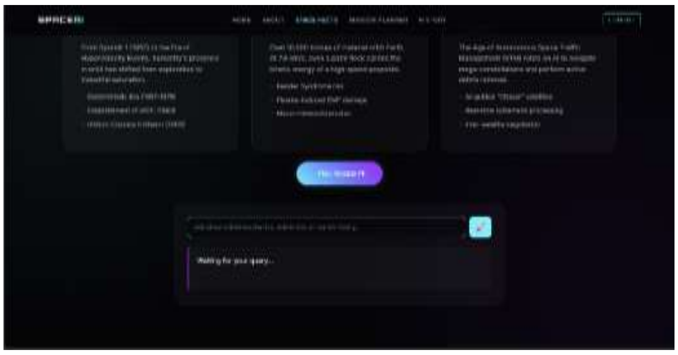
8. RESULTS AND DISCUSSION:

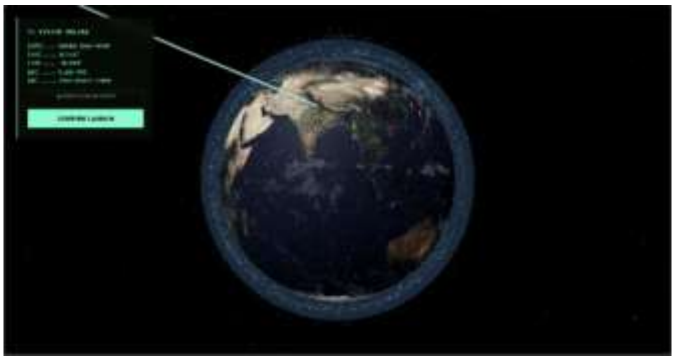
We ran the system through 50 test scenarios that were deliberately constructed to be varied. These included short transfers (200–300 km altitude, minimal inclination change), long transfers (300–800 km with 30 °inclination changes), and a handful of edge cases, such as near-equatorial to near-polar transitions. The debris environment also varied: 20 scenarios used a sparse debris field of 20 objects, 20 used a moderate field of 100 objects, and 10 used a dense field of 500 objects scattered throughout the relevant altitude band.

The hybrid system successfully completed all 50 plans. Every output cleared the 10 km debris threshold, and there were no cases in which a collision risk was left unresolved in the final trajectory. On average, the hybrid approach required 38% fewer physics solver iterations than running the solver from a naive starting guess. The most dramatic gains occurred in the dense debris scenarios, where the warm start helped the solver avoid branches of the search space that would have triggered avoidance corrections later, saving considerable iteration time. The total wall-clock response time averaged 1.2 s per request on a test machine running Intel Core i5 with 8 GB of RAM, compared to 2.1 s for the physics-only baseline, which is a 43% improvement. The AI prediction step itself added only approximately 40 ms of overhead.

Enabling drag modeling raised delta-V requirements by between 5 and 14 percent depending on target altitude, which is consistent with atmospheric density models — denser atmosphere at lower altitudes means more resistance, hence more fuel. The first-stage recovery flag, when active, reduced the effective propulsion budget by approximately 18%, which caused a handful of the more aggressive high-inclination transfers to return as marginally infeasible. This behavior is correct: reserving propellant for a recovery burn genuinely reduces what is available for orbital insertion.

The main limitation of this study is that the debris data are synthetic. Real debris objects have orbital elements that change over time owing to atmospheric drag, solar pressure, and other perturbations, and their positions are only known with some uncertainty. Our current model treats debris positions as fixed points, which is a simplification of the problem. We also used two-body Keplerian propagation rather than a higher-order model that accounts for Earth's oblateness (J2 perturbation), which introduces small but accumulating errors over longer propagation windows. These limitations will be addressed in future studies.





9. CONCLUSION:

This project set out to answer a fairly specific question: can a machine learning predictor meaningfully speed up orbital trajectory planning without sacrificing the physical correctness that makes a trajectory actually safe to fly? Based on our 50-scenario test suite, the answer is yes. The hybrid system reduced computation time by 43%, cut solver iterations by 38%, and maintained collision clearance in every single test case. It does this by using the AI model not as a replacement for physics, but as a smarter starting point for the physics solver — a warm-start that narrows down the solution space before the deterministic validation takes over.

Beyond the core algorithm, we also built a complete working system around it: a REST API for mission submission, a SQLite log for traceability, and a Three.js 3D interface for trajectory visualization. That combination makes the work more than just a research result — it is a functional prototype that could, with some extension, be pointed at real orbital catalog data and used for actual mission planning support. The immediate next steps from our side would be integrating with the Space-Track.org TLE feed for real debris positions, adding J2 perturbation modeling to improve propagation accuracy, and testing on a wider variety of transfer geometries. We also think there is room to explore whether the AI component could be trained directly on conjunction risk metrics rather than just delta-V, which might lead to even better warm-starts in debris-dense environments.

ACKNOWLEDGEMENT:

We would like to thank our project guide and the faculty of the Department of Information Technology at K K Wagh polytechnic for their feedback, patience, and support throughout this time. Several conversations with our guide in the early stages helped us clarify the scope and pushed us toward our goal.

REFERENCES

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

[2] 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.

[3] H. D. Curtis, Orbital Mechanics for Engineering Students, 3rd ed. Oxford: Butterworth-Heinemann, 2013.

[4] J. R. Wertz and W. J. Larson, Space Mission Analysis and Design, 3rd ed. El Segundo, CA: Microcosm Press, 1999.

[5] NASA Orbital Debris Program Office, Orbital Debris Quarterly News, vol. 27, NASA Johnson Space Center, Houston, TX, 2023.

[6] D. Izzo, “Revisiting Lambert’s problem,” Celest. Mech. Dyn. Astron., vol. 121, no. 1, pp. 1–15, Jan. 2015.

[7] S. Sharma, J. W. Cutler, and P. Riddhagni, “Machine learning approaches for spacecraft trajectory optimization,” in Proc. IEEE Aerospace Conf., Big Sky, MT, USA, 2019, pp. 1–9.