AI BASED COLLISION AVOIDANCE SYSTEM

Sampath Guruprasad
Department of Computer Science & Engineering
Reva University
Bengaluru, Karnataka, India
sampathguruprasad@gmail.com

Mrs. Rashmi D
Assistant Professor – Department of Computer Science & Engineering
Reva University
Bengaluru, Karnataka, India
rashmi.d@reva.edu.in

Parikshit Hishiker
Department of Computer Science & Engineering
Reva University
Bengaluru, Karnataka, India
parikshit.hishiker@gmail.com

Shourjya Ghosh
Department of Computer Science & Engineering
Reva University
Bengaluru, Karnataka, India
shourjyaghosh0@gmail.com

Vivek Kumar
Department of Computer Science & Engineering
Reva University
Bengaluru, Karnataka, India
vivek21042004@gmail.com

Abstract*— The sharp rise in the number of active satellites and debris has increased the probability of satellite collision, which is a major threat to the success of space operations. This project proposes the development of an AI-based satellite collision avoidance system that can accurately predict the possibility of satellite collision using machine learning and neural networks. The Two-Line Element (TLE) data set is used to simulate the satellite orbit using standard orbital mechanics equations. For this project we have used different predictive models to suit the needs of the different types of satellite orbits, such as Low Earth Orbit (LEO), Geostationary Earth Orbit(GEO) and finally the Polar Orbits. This is done to factor for the different orbital characteristics of the satellites. The project proposes the development of frontend using opensource library like Cesium JS, which will enable us to visualize the satellites and the avoidance feature in 3D. This system will help in increasing the situational awareness and decision-making process of the satellite operators, furthermore this will also enable students and like to play around with the system.

I. INTRODUCTION

The increasing dependence on satellite-based systems for communication, navigation, Earth observation, and scientific research has led to a rapid growth in the number of active satellites orbiting the Earth. Alongside operational satellites, thousands of defunct satellites, and fragments of space debris occupy various orbital regimes, significantly increasing the risk of accidental collisions. Even a minor collision in space can generate many debris, potentially triggering a cascading effect known as the Kessler Syndrome, which threatens the long-term sustainability of space operations.

Traditional collision avoidance methods rely heavily on deterministic models and manual analysis of conjunction data, which can be time-consuming and limited in their ability to adapt to dynamic and highly congested orbital environments. With the increasing complexity of space traffic, there is a growing need for intelligent, automated systems capable of predicting collision risks with greater accuracy and providing timely decision support to satellite operators.

This project focuses on the development of an AI-based satellite collision avoidance system that leverages machine learning and neural network techniques to predict potential close-approach events. Publicly available Two-Line Element (TLE) data is used to propagate satellite orbits and simulate trajectories across different orbital regimes. Recognizing that satellite behavior varies significantly with altitude and inclination, separate predictive models are designed for Low Earth Orbit (LEO), Geostationary Orbit (GEO), and Polar satellites to improve prediction reliability.

The proposed system integrates orbital data processing, risk prediction, and interactive visualization into a unified framework. A backend architecture handles data ingestion, orbit propagation, and collision probability estimation, while a web-based interface provides real-time visualization and alerts using modern 3D visualization technologies. By combining orbital mechanics with artificial intelligence, this project aims to enhance space situational awareness, support safer satellite operations, and contribute to the sustainable use of near-Earth space. provided. The formatter will need to create these components, incorporating the applicable criteria that follow.

II. METHODOLOGY

A. Selecting Data Acquisition and Preprocessing

The first step will involve in collecting data which is publicly available. Two-Line Element (TLE) datasets from Celestrak which is maintained by the Space Surveillance Network(SSN). The data is then cleaned, formatted in such a manner that it ensures consistency and suitability for the orbital propagation and the machine learning models.

B. Orbit Propagation

We propagate satellite's trajectories over a time period of 48 hours using orbital mechanics and SGP4. Propagation is carried out for the different orbital types, like LEO, GEO and the Polar Orbits, which captures the satellites characteristics in each category.

C. Collision Prediction

In this system we developed three models, each for the orbit types because using a single would give us less accuracy and recall for each type of satellite. Therefore, each model is trained specifically for one type of satellite orbit to improve the system's accuracy and the recall. The models are trained on features such as the relative distance, velocity of the satellite, and orbital parameters.

D. Visualization

The predicted collision risks are shown through the 3D visualization. We are able to display satellite positions, orbits and any potential close-approach events in real time with the help of Cesium JS library.

E. Alert Generation and System Validation

The system will generate real-time alerts for a potential collision event, which is predicted over a time period of 48 hours. We validate the model's predictions, recall and accuracy by comparing data gathered by space-track which provides Conjunction Data Messages (CDM) data.

III. TECHNICAL DETAILS

A. Model Architecture

The OrbitGuard System is designed on a four-layer client-server model. This allows each layer to be independently developed, tested, and deployed. Figure 1 depicts a component diagram of all the system components, including their internal sub-modules, and communication protocols.

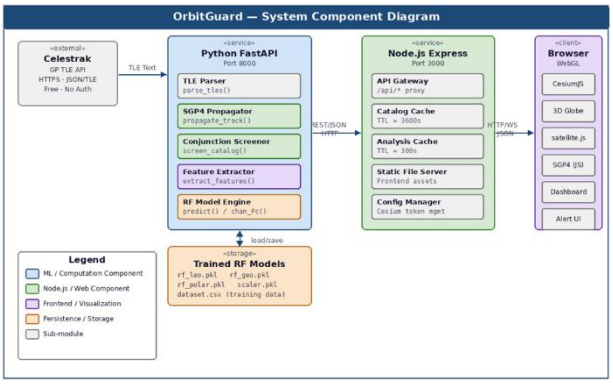


Fig 1.1 System Component Diagram

B. Sequence Diagram

Below Figure presents the sequence of interactions for the primary use case, where a user in the CesiumJS frontend interface selects a satellite and requests a collision analysis for the next 48 hours. It presents the message flow between all seven participants, from the browser to Node.js, Python, the Celestrak external API, and back, in the format specified by the UML 2 notation for sequence diagrams. Dashed arrows indicate the response messages, while the 'opt' fragment refers to a conditional action that takes place if the catalog cache has expired.

When a user selects the Analyze button in the CesiumJS frontend, passing in a satellite NORAD ID, such as that of the ISS (25544), a request is made to a Node.js gateway service. This service, in turn, caches the information or, failing that, calls a Fast API service written in Python. This service ensures that its satellite catalog is current, parses it, and uses a propagator to compute satellite position tracks for all satellites over a period of 48 hours. A conjunction screen is then used to filter potential close approaches, resulting in candidate events. These events are then passed to an RF model, resulting in collision probabilities and risk classification. These are then serialized back to JSON, passed back to the Node.js service, and then back to the front end, where a CesiumJS dashboard displays conjunction information, charts, and warnings for high-risk events.

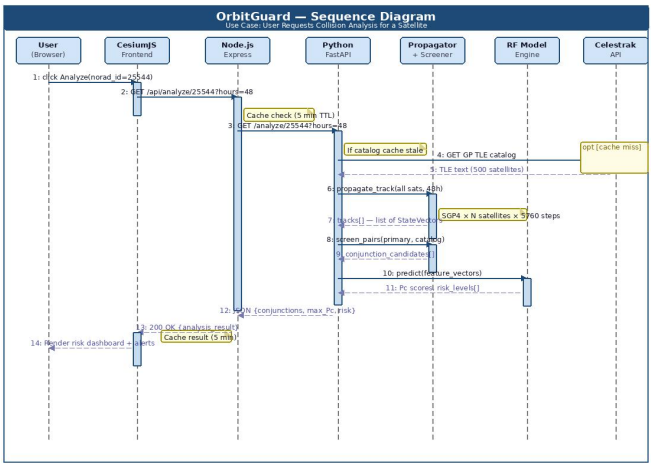


Fig 2.1 Sequence Diagram

C. Entity Relationship Model

The ER diagram for the "OrbitGuard" data domain is presented in the figure below. Although the current prototype uses in-memory storage and CSV files to hold its data, the ER model describes the logical data entities, attributes, data types, and relationships between them. This model can be used as a blueprint to design a relational database schema in the future, using a DBMS like PostgreSQL or SQLite.

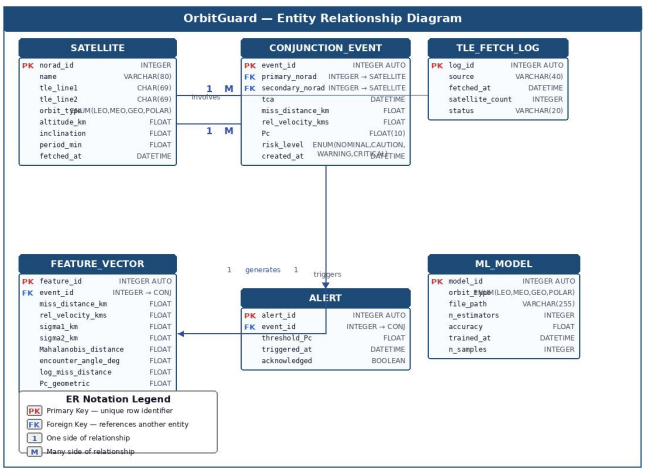


Fig 3.1 ER Diagram

IV. SOFTWARE REQUIREMENTS

A. ***SGP4(Simplified General Perturbations-4):***

The SGP4 is a well-known mathematical model and algorithm for calculating the position and velocity of Earth-orbiting satellites. It provides a standardized way of processing the position and velocity of satellites over a period. It is compatible with various programming environments like Python, C++, and MATLAB. It is the primary algorithm for processing the Two-Line Element Sets of various organizations like NORAD. It is widely used in various applications like satellite tracking and orbit prediction because of the balance between computational speed and accuracy. It can accurately compute the position and velocity of the satellite in real-time by incorporating various complex environmental factors like the non-spherical shape of the Earth, atmospheric drag, and gravitational attraction of the Moon and Sun.

B. ***Fast API + Uvicorn:***

The FastAPI + Uvicorn is a well-known high-performance web framework and server combination for developing APIs using Python, offering a variety of tools for rapid development, documentation, and asynchronous development. It is compatible with modern Python features, including type hints and asyncio, and supports standard web protocols such as HTTP, WebSocket, and GraphQL. FastAPI is widely used in developing machine learning model deployment, real-time data streaming, and microservices due to its high performance, comparable to Node.js and Go, and user-friendly features such as automatic Swagger UI generation. FastAPI is easy to integrate with Pydantic for data validation, supports a variety of database engines, and authentication systems.

C. ***CesiumJS:***

Cesium JS is one of the most popular open-source JS library for the development of 3D globes and 2D maps. It also offers a wide variety of tools and techniques for visualization, analysis and showing geospatial data. This library has a extensive community and is compatible with web browsers such as Chrome, Firefore, Safari and Edge. In terms of data formats, the library is compatible with various formats such as 3D Tiles, KML, and GeoJSON. It is widely used in various applications such as satellite tracking, urban planning, and defense simulation. This is possible because of the high-precision rendering and feature set of the library. Furthermore, this library offers us cloud services like Cesium ion, and it also offers hardware-accelerated graphics using WebGL.

D. ***Python:***

Python is an object oriented, high-level programming language known for its simplicity, readability. It is beginner-friendly which makes it a popular choice for new programmers. It is commonly used in for a wide range of applications like data analysis, machine learning, artificial intelligence, web development, automation, and scientific computing. Due to it's flexible nature it allows developers to use the same language for various purposes and for simplifying the development. process.

Python's key advantages lies in its wide and dense libraries, which includes modules and functions that make tasks easier. This allows us to achieve significant functionality with minimal code and our saving time and effort. Furthermore, python is able to support object-oriented and functional programming which gives the developers the freedom to select the approach which best suits their project requirements.

E. ***VS Code:***

Microsoft created the free source-code editor known as Visual Studio Code, or VS Code for short. Along with features like debugging, syntax highlighting, code completion, and Git integration, it supports many programming languages. Moreover, users can customize the editor to suit their preferences thanks to its customizable user interface. The reason for VS Code's widespread use among developers of all skill levels is it's lightweight design quick performance, and wide selection of extensions. Due to its cross-platform interoperability, it may be utilized on Linux, macOS, and Windows.

V. EXISTING SYSTEM

Currently, a few government-run systems dominate the satellite conjunction screening operational landscape. Despite their great potential, these systems are limited by classification requirements, limited data access, and architectures that were created for catalog sizes that predate the mega-constellation era. The Space Surveillance Network (SSN) of the United States Space Force (USSF) is the main source of orbital tracking data for conjunction analysis. This system is made up of a global network of about 30 optical telescopes and ground-based radars that can track objects in Low Earth Orbit (LEO) that are larger than about 10 cm. Conjunction Data Messages (CDMs) are sent to registered satellite operators via the Space-Track portal, which distributes catalog data from the SSN. Complete 6x6 position-velocity covariance matrices are included in these CDMs, allowing for exact probability of collision (Pc) calculations. However, access is restricted for universities, small businesses, and researchers in many areas because it requires formal approval, registration, and agreement to terms of service. One of the earliest automated conjunction screening services available to the public was SOCRATES, which was hosted by Celestrak. It ranks close approaches according to miss distance and probability of collision (Pc), conducts catalog-wide screening, and releases daily reports of the most significant conjunction events. The system uses standard Pc estimation methods and depends on Two-Line Element (TLE) data from the SSN. However, rather than providing on-demand analysis for satellites, SOCRATES functions as a batch-processing system with a set daily update cycle. Furthermore, it lacks a programmatic access API.

The lack of covariance data in TLE-based computations is another drawback; this makes uniform sigma assumptions necessary and lowers the precision of Pc estimates in comparison to covariance-based techniques.

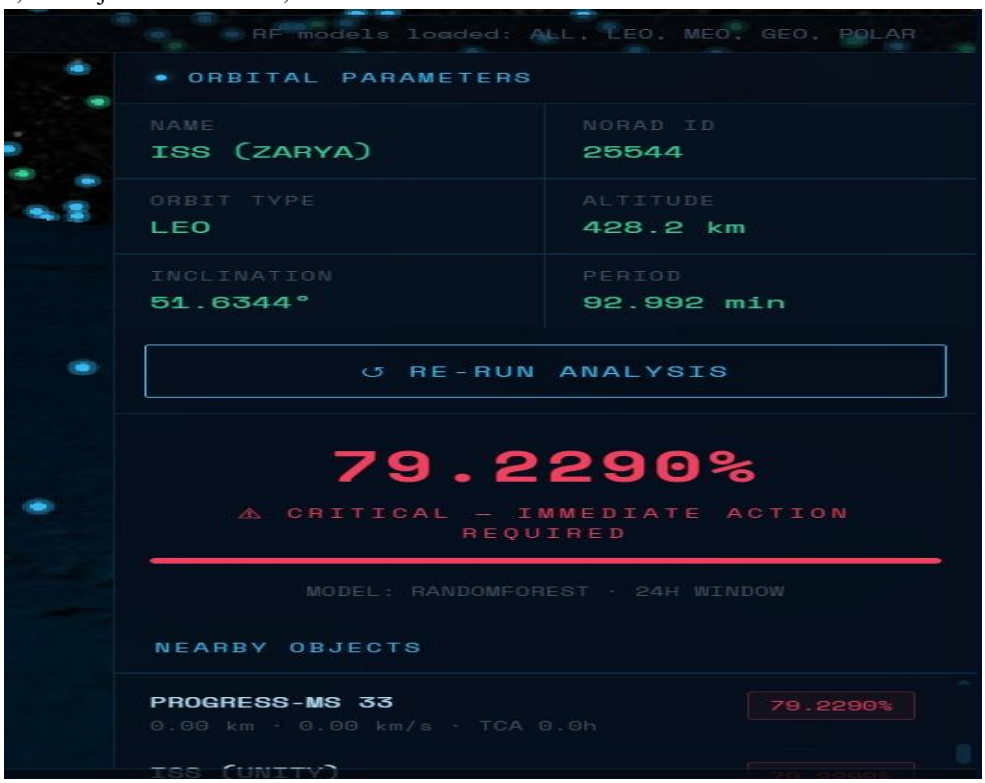
Through its Space Debris Office at the European Space Operations Centre (ESOC) in Darmstadt, the European Space Agency (ESA) provides a conjunction assessment service. This service uses the Space-Track infrastructure to distribute CDMs after screening ESA satellites against a catalog of about 23,000 tracked objects. For every

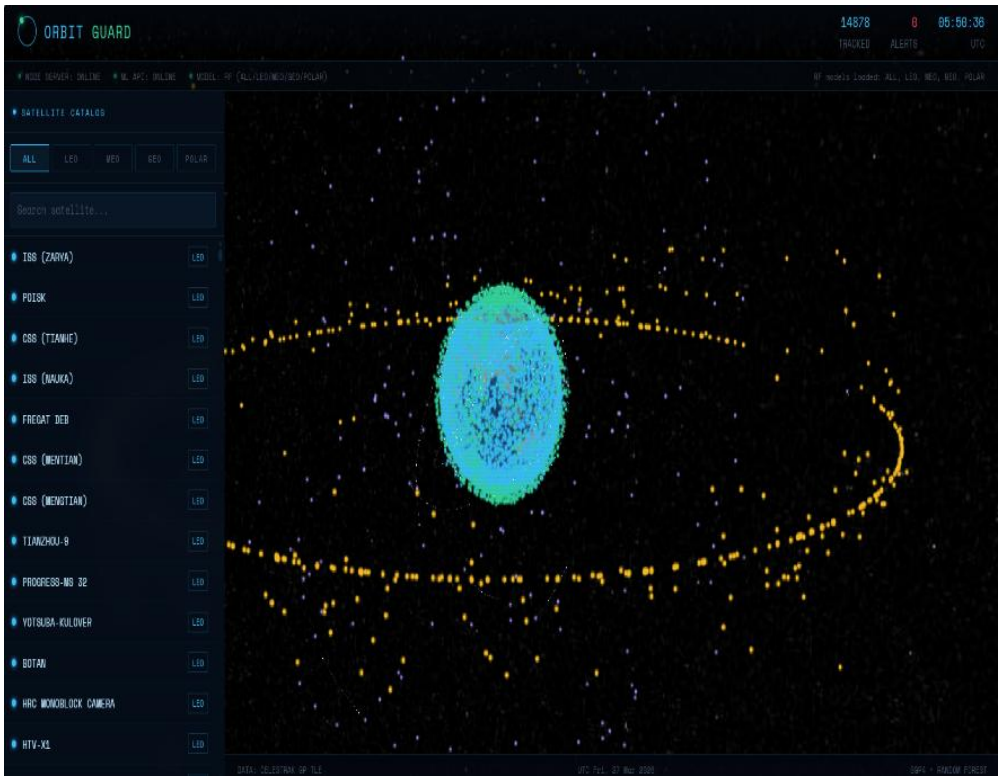
operational satellite, the system generates a significant number of conjunction alerts every week. When the likelihood of a collision surpasses the operational threshold, maneuver decisions are usually made. Nevertheless, this service is not available to the public and is limited to ESA missions. Additionally, as the number of tracked objects increases, the workflow's heavy reliance on operator manual triage becomes unsustainable.

NASA spacecraft are subject to conjunction screening by the Conjunction Assessment Risk Analysis (CARA) team. High-fidelity orbit propagation models, like the High Precision Orbit Propagator (HPOP), which employs atmospheric density models powered by real-time solar flux data, are incorporated into the CARA workflow. Using predetermined probability thresholds, this method facilitates operational decision-making and allows for extremely accurate conjunction predictions. Despite its complexity, CARA's methods and implementation specifics are not made public, and it is not available to external users. Several platforms have emerged that seek to bridge this gap. Some of these platforms include Exo Analytic Solutions, Slingshot Aerospace, and COMSPOC. These platforms offer higher tracking rates, improved detection limits, and, in some cases, API-based solutions. However, these platforms are priced in a manner that is inaccessible to academic users, given that they cost several thousand dollars a month. Furthermore, these platforms do not offer transparency in terms of their algorithms and models. Within both governmental and commercial domains, there are four major limitations. First, there is an issue of accessibility. There is an issue of classification restrictions, registration difficulties, and cost. These issues are especially problematic for universities, new organizations, and independent researchers. Second, there is an issue of scalability. Most existing systems were originally developed to accommodate catalogs consisting of only 20,000 to 30,000 objects. In addition, with the advent of mega-constellations, it is expected that catalog sizes will be greater than 100,000 objects, thus greatly increasing complexity. Third, there is an issue of transparency. Most operational systems do not reveal information regarding their algorithms and models. Fourth, there is an issue of real-time capabilities. Most existing systems, such as SOCRATES, only operate on a set schedule. They do not have real-time capabilities. They do not allow on-demand conjunction analysis. The challenges that are being experienced highlight a critical gap within the existing space situational awareness ecosystem. This is because, as the space environment becomes increasingly congested and contested, the need for tools that can facilitate conjunction analysis in an accessible, scalable, and transparent manner is increasingly urgent. This is particularly true given the growing stakeholder community within the space environment, including private industry and new entrants such as academic organizations and new space-faring nations that need access to effective solutions that are cost-efficient. If such limitations are not addressed, there is a greater probability that there could be negative impacts on the sustainability of space activities.

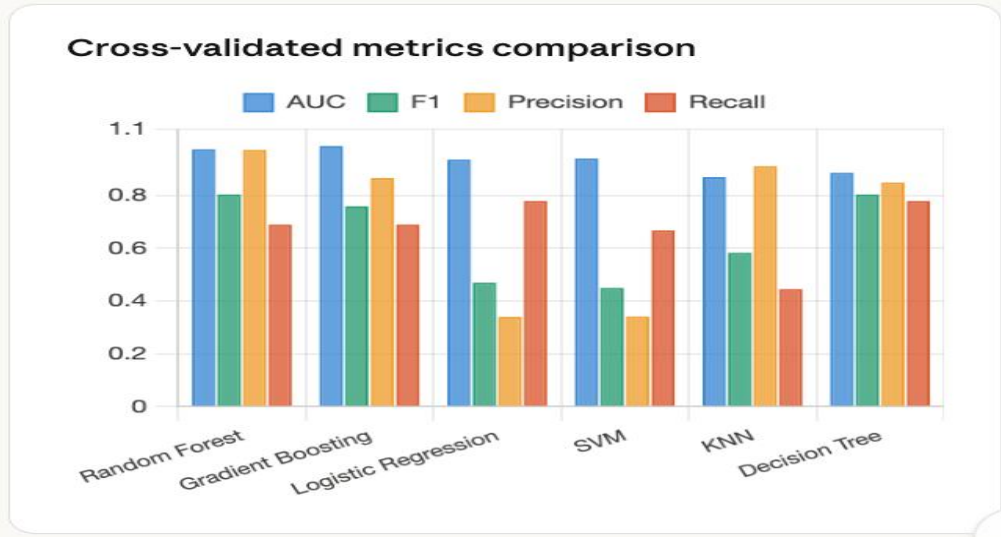
OrbitGuard is designed to address these issues by providing:
(1) Accessible and open conjunction screening,
(2) real-time and on-demand analytical tools,
(3) access through API for developers and researchers,
(4)transparent and reproducible methodologies. All these factors combined establish OrbitGuard's utility and relevance as an educational and research tool in the field of space situational awareness.

VI. RESULTS




ORBIT GUARD
14878
TRACKED
0
ALERTS
05:50:38
UTC
SATELLITE CATALOG
ALL
LEO
MEO
GEO
POLAR
Search satellite...
ISS (ZARYA)
POISK
CSS (TIANHE)
ISS (NAUKA)
FREGAT DEB
CSS (WENTIAN)
CSS (MENGTIAN)
TIANZHOU-9
PROGRESS-MS 32
YOTSUBA-KULOVER
BOTAN
HRC MONOBLOCK CAMERA
HTV-X1


ORBIT GUARD
14878
TRACKED
0
ALERTS
05:51:06
UTC
SATELLITE CATALOG
MEO
LCS 1
LCS-5
LAGEOS 1
PHASE 3D (AO-10)
COSMOS 1000 (ETALON 1)
COSMOS 2024 (ETALON 2)
LAGEOS 2
RADIO ROBIG (RBIG)
POLAR
NAVSTAR 43 (USA 132)
NAVSTAR 46 (USA 145)
IMAGE
NAVSTAR 48 (USA 151)


Cross-validated metrics comparison
AUC
F1
Precision
Recall
1.1
0.8
0.6
0.4
0.2
0
Random Forest
Gradient Boosting
Logistic Regression
SVM
KNN
Decision Tree

VII. CONCLUSIONS

In conclusion, Orbit Guard shows that open-source tools and publicly accessible orbital data can be used to create a complete, ML-augmented satellite collision avoidance prototype. The system achieves all five of the Phase-I report's goals: data collection from Celestrak, orbit propagation based on SGP4, AI-driven collision probability estimation, automated alert generation, and 3D interactive visualization. It shows the feasibility of the AI-based method for use by space agencies, satellite operators, and research institutions and offers a strong basis for a production-quality space traffic management tool. Future research can go in a few different directions. First, since Space-Track Conjunction Data Messages (CDMs) contain operator-verified close approaches and full covariance matrices, substituting actual conjunction data from CDMs for the Celestrak TLE-based training labels would greatly increase model accuracy. Second, the decision support loop would be completed by incorporating a maneuver recommendation engine the existing system identifies and measures risk but does not recommend preventative measures.

VIII. REFERENCES

[1] A. A. I. Siddiqui, "The integration of AI technologies in space traffic management," Bentham Direct, Open Astron. J., vol. 18, pp. 1–12, Mar. 2025.

[2] S. Schaefer and M. G. Cichon, "Autonomous trajectory optimization and collision management using neural systems," IEEE Eng. Manag. Rev., vol. 53, no. 1, pp. 88–101, Jan. 2025.

[3] C. Ieracitano, N. Mammone, P. Lanza, and F. Gao, "AI for space: theories, models and applications," Neural Comput. Appl., Springer, New York, NY, USA, vol. 37, pp. 1–22, Feb. 2025.

[4] T. S. Kelso, "Frequently asked questions: Two-line element set format," CelesTrak, Colorado Springs, CO, USA, Tech. Note, 2019. [Online]. Available: https://celestrak.org/columns/v04n03/

[5] J. L. Foster and H. S. Estes, "A parametric analysis of orbital debris collision probability and maneuver rate for space vehicles," NASA Johnson Space Center, Houston, TX, USA, NASA Tech. Memo. 100559, Aug. 1992.

[6] R. P. Patera, "General method for calculating satellite conjunction probability," J. Guid. Control Dyn., AIAA, Reston, VA, USA, vol. 24, no. 4, pp. 716–722, Jul. 2001.

[7] F. R. Hoots and R. L. Roehrich, "Models for propagation of NORAD element sets," Spacetrack Report No. 3, U.S. Air Force Aerospace Defense Command, Colorado Springs, CO, USA, Dec. 1980.

[8] NASA Orbital Debris Program Office, "Orbital debris quarterly news," NASA Johnson Space Center, Houston, TX, USA, Tech. Rep., vol. 28, no. 1, Feb. 2024.

[9] L. Breiman, "Random forests," Mach. Learn., Springer, New York, NY, USA, vol. 45, no. 1, pp. 5–32, Oct. 2001.