SDCRA: A Web-Based Multi-Agent Framework for Real-Time Space Debris Collision Avoidance and Visualization

Arpit Sharma, [Team Member Names]
Department of Information Science & Engineering
B.M.S. College of Engineering
Bengaluru, India
Email: arpitsharma@email.com

***Abstract*—The exponential increase in Low Earth Orbit (LEO) satellite constellations has intensified the risk of the Kessler Syndrome—a cascading chain of orbital collisions. Existing Space Situational Awareness (SSA) tools are often proprietary, computationally expensive, or lack accessible visualization interfaces. This paper proposes the Space Debris Collision Risk Assessment (SDCRA) system, a web-based framework that democratizes orbital analysis. SDCRA utilizes the SGP4 propagation model directly within the client browser to simulate satellite trajectories in real time. The system integrates a Three.js accelerated rendering engine to visualize orbital conjunctions and employs a Multi-Agent System (MAS) architecture to simulate collision avoidance maneuvers. Performance benchmarks demonstrate the system's ability to propagate and render over 5,000 debris objects at 60 FPS on standard hardware.**

***Index Terms*—Space Debris, SGP4, WebGL, Collision Avoidance, Multi-Agent Systems, React.js, Orbital Mechanics**

I. Introduction

Space debris poses a critical threat to global telecommunications, navigation, and Earth observation infrastructure. With over 27,000 tracked objects and millions of smaller untracked fragments, the probability of collision events continues to rise.

Traditional SSA systems rely heavily on server-side batch processing, introducing visualization latency. SDCRA addresses this limitation by shifting orbital propagation and visualization to the client side using optimized JavaScript and WebGL rendering.

The primary contributions of this work include:

- A real-time SGP4-based physics engine executed in the browser.
- A dynamic conjunction assessment module computing Time of Closest Approach (TCA).
- A high-performance visualization layer using instanced rendering.
- A modular Multi-Agent simulation framework for decentralized collision avoidance.

II. Mathematical Modeling

A. SGP4 Propagation

The SGP4 propagator accounts for secular and periodic perturbations caused by Earth's oblateness (J_2, J_3, J_4 zonal harmonics) and atmospheric drag.

Given a Two-Line Element (TLE) set at epoch t_0, the satellite state vectors are computed as:

$$\vec{r}(t), \vec{v}(t) = \text{SGP4}(TLE, t) \tag{1}$$

where $\vec{r}(t)$ and $\vec{v}(t)$ represent position and velocity vectors in Earth-Centered Inertial (ECI) coordinates.

B. Conjunction Assessment

Collision risk is determined by the Euclidean separation distance:

$$d_{\text{miss}}(t) = \|\vec{r}_{\text{target}}(t) - \vec{r}_{\text{debris},j}(t)\| \tag{2}$$

A high-risk alert is triggered when:

$$d_{\text{miss}} < D_{\text{threshold}} \tag{3}$$

For simulation purposes:

$$D_{\text{threshold}} = 50 \text{ km}$$

III. System Architecture

The SDCRA system follows a modular layered design.

A. Data Ingestion Layer

- Fetches live TLE data from CelesTrak and Space-Track.org via REST APIs.
- Parses raw TLE strings into classical orbital elements.

B. Physics Core

- Implemented using the `satellite.js` library.
- Executes propagation at 1 Hz for all active objects.
- Converts ECI coordinates to geodetic form for visualization.

C. Visualization Layer

- Built using React Three Fiber (R3F).
- Uses `InstancedMesh` for efficient rendering of thousands of debris objects.
- Maintains stable frame rates near 60 FPS.

D. Interface Layer

- Implements a Glassmorphism-based HUD interface.
- Includes secure user session handling.
- Provides a "Hold-to-Engage" thruster interface for maneuver simulation.

IV. Implementation Details

The system is developed using a MERN-stack variant with a React/Vite frontend. A key innovation is the safety-inspired maneuver trigger mechanism that reduces accidental activation during simulations.

A. SGP4 Propagation Loop

```
const getSatPosition = (tleLine1, tleLine2) => {
  const satrec = satellite.twoline2satrec(tleLine1,
      tleLine2);
  const positionAndVelocity = satellite.propagate(
    satrec,
    new Date()
  );

  return satellite.eciToGeodetic(
    positionAndVelocity.position,
    satellite.gstime(new Date())
  );
};
```

V. Simulation and Results

A. Performance Benchmarking

Tests were conducted on a standard laptop (Intel i5, integrated GPU).

- Object Count: 5,000 debris objects
- Frame Rate: Stable at 58–60 FPS
- GPU Memory Usage: Approximately 200 MB VRAM

B. Case Study: ISS Conjunction

A simulated conjunction event between the ISS (ZARYA) and a mock debris object (ID: 49044) was successfully detected. The system computed a TCA of less than 22 minutes and triggered a critical alert state within the dashboard.

VI. Future Scope

Future extensions include integrating Reinforcement Learning-based autonomous agents. Each satellite will function as an independent decision-making agent capable of negotiating collision avoidance maneuvers within a decentralized SSA environment.

VII. Conclusion

SDCRA demonstrates that high-fidelity orbital propagation and conjunction visualization can be achieved entirely within a web browser. By lowering accessibility barriers to SSA tools, SDCRA contributes toward improved education, research, and sustainable space operations.

References

[1] D. J. Kessler and B. G. Cour-Palais, "Collision frequency of artificial satellites: The creation of a debris belt," *Journal of Geophysical Research*, 1978.

[2] T. S. Kelso, "CelesTrak: Orbit Visualization and Analysis," 2023. [Online]. Available: https://celestrak.org

[3] D. A. Vallado, *Fundamentals of Astrodynamics and Applications*, Microcosm Press, 2013.