UAV Deconfliction system

Name: Rugved Tamhan

BTech in Robotics and Automation

The purpose of this project was to build a robust and modular system capable of evaluating whether a planned drone mission would result in a conflict with other drones already operating in a shared airspace. In increasingly automated and crowded environments, managing drone traffic with safety and precision is vital. This project focuses on detecting spatial and temporal conflicts — also known as 4D conflicts — involving position (x, y, z) and time (t). The system simulates drone movement over time, compares their positions, and determines whether they come too close to each other during overlapping time windows.

The core of the system is built in Python using a modular architecture. It consists of separate components for mission data loading, position interpolation, conflict detection, conflict classification, and visualization. The mission data, including all waypoints, are stored in a structured JSON file. Each drone's path is defined by a series of 3D waypoints with timestamps. These waypoints are linearly interpolated to simulate drone movement at each second of their mission. By simulating motion at each discrete time step, the system can continuously evaluate the distance between the primary drone and all others in flight during that period.

When the system finds that the distance between two drones drops below a predefined safety threshold (2.5 meters), it records a conflict. But rather than just reporting a simple "conflict found," the system goes further by classifying the type of conflict. There are multiple possible types: a full 4D conflict where position and time both overlap, altitude-only conflicts where x/y/t are the same but altitude differs slightly, time-only conflicts where two drones are flying at the same time but in totally different areas, and spatial-only conflicts where the positions overlap but at different times. This classification adds an intelligent, explainable layer to the system that helps the user understand not just *if* there's a problem, but *what kind* of problem it is.

I tested the system with five different primary drone missions, including one that was fully safe and four that deliberately conflicted with other drones in different ways. These missions interacted with a set of nine other drones flying diverse paths. I also considered edge cases like hovering drones (only one waypoint), drones flying near but not overlapping, and paths that share time windows without physical proximity. The conflict classification system correctly identified and labelled each case, and the outputs were verified both through console reports and 3D visualizations.

The visualizer uses matplotlib to render a 3D representation of the drone flight paths. The primary drone is shown in blue, other drones in grey, and any detected conflict point is marked in red. This gives users an instant, intuitive understanding of when and where a conflict occurs. All of this is printed alongside a structured conflict report detailing the conflict time, distance, position, involved drone ID, and the type of conflict detected.

While the system currently uses rule-based logic for classification and resolution suggestions, there's clear scope for AI integration in the future. A machine learning model, trained on simulated conflict data, could predict the likelihood or type of conflict based on path characteristics. Reinforcement learning could be applied to dynamically suggest or even generate new paths that avoid congested areas. The system could also evolve to include real-time drone telemetry, integration with external simulators like Gazebo or AirSim, or a user-friendly GUI to drag, drop, and edit waypoints interactively.

Through this project, I gained hands-on experience with simulating real-world dynamic systems in 4D, writing modular and reusable code, and building visual tools to interpret simulation results. It taught me to consider practical edge cases, scalability, and clarity in outputs. Most importantly, it gave me insight into the kind of problems engineers face in building autonomous airspace management systems.

In conclusion, the UAV Deconfliction System successfully fulfils its core objective of checking whether a mission is safe to fly in shared airspace. It supports multiple drones, identifies conflicts accurately, classifies them meaningfully, and visualizes the results in a user-friendly way. It is extensible, practical, and lays the groundwork for more intelligent and scalable drone traffic management systems in the future.