README: UAV Conflict Detection and PPO-Based Deconfliction Simulation

Overview

This project simulates the movement of autonomous UAVs (drones) through a shared 3D space. Each UAV is trained using Proximal Policy Optimization (PPO) to navigate from a start point to a goal while avoiding conflicts with other UAVs.

Requirements

- Python 3.8+
- pip (Python package installer)

Dependencies

Install required libraries via pip:

```
pip install numpy pandas plotly stable-baselines3 gymnasium
```

How to Run

1. Save the provided Python code in a file (e.g., `drone_simulation.py`).
2. Run the file using:

```
python drone_simulation.py
```

3. The simulation will:
   - Train each UAV agent using PPO
   - Detect conflicts using Euclidean distance checks
   - Generate a 3D animation and conflict trigger graph
   - Output a CSV log of drone trajectories

Output

- `drone_position_log.csv`: Time-series log of UAV positions
- 3D animation of UAV flight paths
- Real-time spike graph showing conflict events
- Time-vs-Position plots for each axis

Reflection & Justification Document

Design Decisions & Architecture

The architecture separates concerns across distinct phases:

- **Environment Design**: The `DroneEnv` class models a single UAV environment using `gymnasium`. Each step includes positional updates, time increments, and reward shaping.
- **Training**: Each UAV is trained independently for multiple episodes using PPO. The best-performing model (based on total reward) is selected.
- **Conflict Detection**: After all paths are generated, a spatial check loop iterates through each timestep to find conflicts between UAVs.
- **Visualization**: 3D animations, conflict triggers, and position plots are rendered using Plotly.

Spatial & Temporal Conflict Checks

- **Spatial**: Conflicts are identified if two UAVs are within a Euclidean distance of `4.0` units at the same timestep.
- **Temporal**: Checks occur at discrete time steps (`t = 0` to `time_limit = 80`). Drones' time progressions are based on learned PPO behavior.

AI Integration

- **PPO (Proximal Policy Optimization)** from Stable Baselines3 is used for path planning.
- Each drone has its own instance of the PPO agent trained within a gym environment.
- The model encourages goal-directed motion while discouraging delays and near misses.

Testing Strategy & Edge Cases

- **Training Episodes**: Each drone is trained over multiple episodes, with the best model selected.
- **Edge Cases**: The system can handle overlapping start/goal locations, path convergence, and extreme delays.
- **Data Validity**: Conflict logs are validated to ensure they contain the required columns (`time`, `drone_1`, `drone_2`).

Scalability Considerations

To scale for real-world UAV traffic (e.g., 10,000+ drones):

- **Parallelization**: Distribute training and inference across multiple CPU/GPU nodes.
- **Event-Driven Simulation**: Replace timestep loop with an event-queue-based simulation to process asynchronously.
- **Vectorized Operations**: Use NumPy broadcasting or GPU acceleration (e.g., with CuPy or JAX) for conflict detection.
- **Conflict Zones**: Use spatial partitioning (e.g., Octrees or k-d trees) to reduce conflict pair checks from O(n^2) to O(n log n).
- **Streamed Data Handling**: Transition to a real-time data pipeline using tools like Apache Kafka or ROS2.

Future Work

- Integration with air traffic control APIs or real sensor feeds
- Dynamic re-routing via multi-agent reinforcement learning
- No-fly zone handling and weather-aware navigation