Drone Deconfliction & Visualization System

Simulation, AI Prediction, and Conflict Resolution in 4D Space

Name : Praful Navnath Pawar(BE in Ai and Data Science)

A Working System in Action

Mission Input

- Waypoints and other drone paths are stored in lists or arrays.
- Safety distance is a constant used during conflict checks.

Data Preprocessing

- interpolate() function is used to find exact positions at each small time step.
- This ensures fairness when checking drone positions together.

. AI Prediction Engine

- predict() function uses simple math to guess the next position.
- Prediction is done for **every time step** ahead.

Conflict Detection Engine

- Loops compare every drone's position at each time.
- Safety rules are applied here to decide if it's risky.

Decision Layer

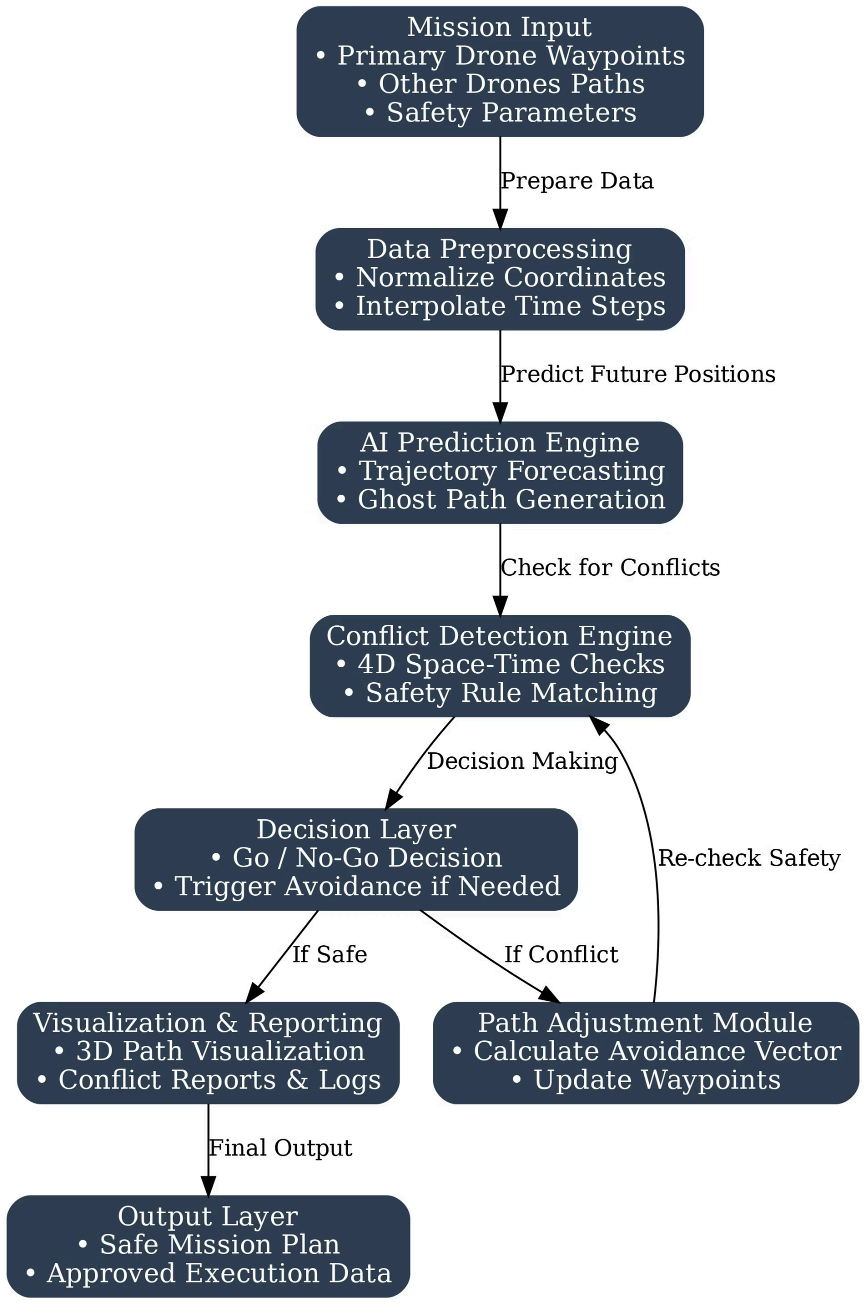
- Conditional checks decide if we need to adjust the route.

Visualization & Reporting

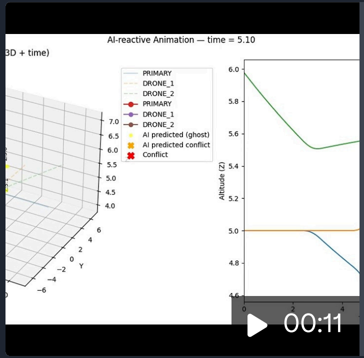
- visualize_conflicts() uses Matplotlib to draw everything in 3D

Output Layer

- Final updated waypoints are stored and can be saved/exported.



Conflict Scenario:



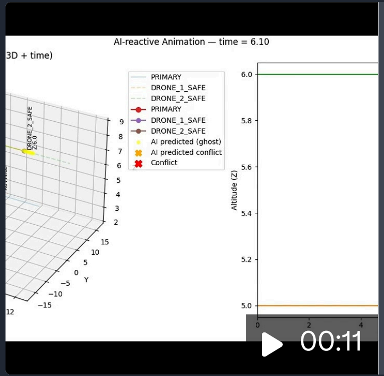
YouTube

conflict ai reactive

aconflicts in drone druing flying and prdective Ai predicts before it...

- **AI Prediction** →
- Uses functions like `predict_future_positions()` in **DroneAIPredictor**.
- Predicts each drone's path for the next few seconds using **linear fit** math.
- This generates "ghost positions" (future spots).
- **Conflict Check** →
- In **DroneConflictDetector**, function like `check_conflict()` loops over drones in pairs.
- For each predicted time step:
  - Measures **3D distance** between drones.
  - If the distance < **safety threshold**, marks it as a potential conflict.

Conflict-Free Scenario



YouTube

no conflict ai reactive

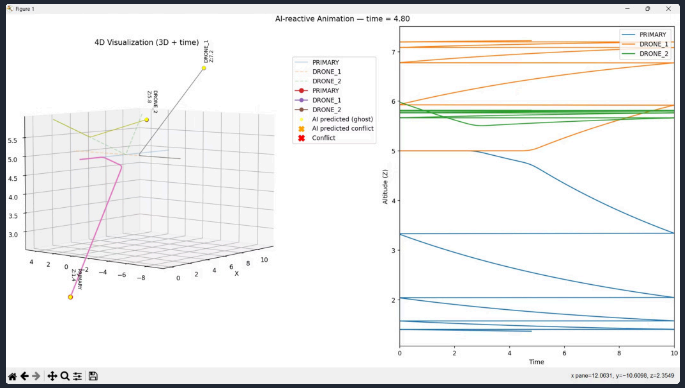
this video showcase their is not happend any conflicts drone move...

.**Detect conflict point** (time & location where collision would happen).

- **Plan avoidance maneuver**:
- Adjusts **altitude** (Z-axis change) or slightly changes X/Y path.
- Makes sure the new route keeps the drone above the safety distance.
- **Update path**:
- Safe route stored as `_SAFE` path (e.g., `DRONE_1_SAFE`).
- These are then drawn in the simulation instead of the original risky route.

4D Visualization Feature:

.



Primary vs Safe Paths

Original planned route = **solid lines** (PRIMARY, DRONE_1_SAFE, etc.).

Adjusted safe route = **dashed lines** (_SAFE suffix).

AI Predicted Ghost Positions

Yellow markers = where AI thinks a drone will be **in the near future**.

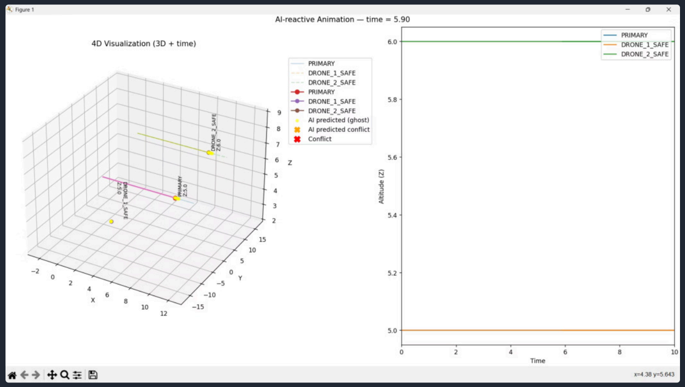
Helps visualize AI's "look-ahead" capability.

Conflict Markers

- **Red "X"** = where the AI predicts a collision could happen.
- Appears in the same frame as the predicted conflict time.

```
ax.scatter(conflict_x, conflict_y, conflict_z, color='red', marker='x', label='Conflict')
```

.



Dynamic Labels

Drone IDs & altitudes printed next to drones in the 3D plot.Updates each frame → feels interactive.

```
ax.text(x, y, z, f"{drone_id}\nZ={altitude:.1f}", fontsize=8)
```

Real-Time Animation

- Uses **Matplotlib's** FuncAnimation to continuously update the figure.
- Every few milliseconds, the plot refreshes → giving smooth movement.

```
ani = FuncAnimation(fig, update_function, frames=num_frames, interval=100)
```

Side Graph (Z vs Time)

- Right-hand graph shows **altitude changes** for each drone over time.
- Lets you track **how avoidance maneuvers happen**.

Predective Ai Integration:

a)Input Gathering

```
history = drone.get_recent_positions()  velocity = drone.get_current_velocity()
```

b) Prediction

```
future_pos = current_pos + velocity * lookahead_time
```

c) Conflict Detection (on Predicted Positions)

```
if distance(future_pos1, future_pos2) < safety_distance: predicted_conflict = True
```

d) Resolution

```
avoidance_vector = compute_offset(drone1, drone2) adjusted_velocity = base_velocity + avoidance_vector
```

e) Feedback to Visualization

- **Yellow ghost markers** show AI's predicted positions.
- **Red conflict points** show where AI expects problems.
- Adjusted safe paths appear as **dashed or curved lines** in your animation.