STRATEGIC DECONFLICTION IN AIRSPACE

DESIGN APPROACH:

Focusing on building a strategic algorithm for deconfliction of drones in airspace by executing a waypoint mission for single drone (i.e. primary mission drone) along with multiple drones (two in this algorithm). Drone's route is predefined and hard coded. Initial query is given by the drone's way point algorithm. Considering the functionality wise verification and validation, intentionally some of the time stamps of drone 1 and 2 are designed to be conflicted with prime mission to show the capability of algorithm to sort the deconfliction issues, such as in this algorithm design the functional aspects of **spatial check, temporal check, conflict explanation and query interface** is focused.

ALGORITHM FLOW:

Step 1: Inputs the hard coded way point data for primary mission and other two drones.
Step 2: Verifying and validating query based command at starting.
Step 3: Plotting 3 dimensional way points of all three drones (primary mission drone and other two drones).
Step 4: Intentionally creating conflicts, to display the occurrence of conflicts and information of time, position and drone number.
Step 5: Displaying all the conflicts data and 3-D graph is used for visualization of conflicts (a colour changing action is observed on conflicting of one drone's position with another.

MATLAB CODE:

```
%% =========================
% 3D Multi-Drone Simulation with Query Interface and Real-Time Conflict Animation
clc; clear; close all;


%% ======== Simulation Parameters ========
T_start = 0; T_end = 20; N_points = 200; dt = 0.1;
safetyBuffer = 10; % meters


%% ======== Primary Drone Path ========
theta = linspace(0,2*pi,N_points);
x_main = 50*cos(theta);
y_main = 30*sin(theta);
z_main = 15*sin(2*theta)+50;
t_main = linspace(T_start,T_end,N_points);


%% ======== Other Drone Paths ========
% Drone 1
otherDrones(1).x = linspace(-60,60,N_points);
otherDrones(1).y = linspace(-40,40,N_points);
otherDrones(1).z = 50 + 5*sin(linspace(0,4*pi,N_points));
otherDrones(1).t = linspace(T_start,T_end,N_points);


% Drone 2
```

```
otherDrones(2).x = 40*cos(theta);
otherDrones(2).y = 20*sin(theta);
otherDrones(2).z = 55*ones(1,N_points);
otherDrones(2).t = linspace(T_start,T_end,N_points);


% Drone 3
otherDrones(3).x = 30*cos(theta);
otherDrones(3).y = 15*sin(theta);
otherDrones(3).z = 52*ones(1,N_points);
otherDrones(3).t = linspace(T_start,T_end,N_points);


%% ======== Query Interface Check Before Animation ========
[status, conflictDetails] =
queryMission(x_main,y_main,z_main,t_main,otherDrones,safetyBuffer);


disp("====== Mission Query Interface ======");
disp("Mission Status: " + status);
if ~isempty(conflictDetails)
disp("Conflict Details:");
disp(conflictDetails);
else
disp("No conflicts detected. Primary mission path is safe.");
end


%% ======== Visualization Setup ========
figure('Color','k'); hold on; grid on; axis equal;
xlabel('X','Color','w'); ylabel('Y','Color','w'); zlabel('Z','Color','w');
set(gca,'Color','k','XColor','w','YColor','w','ZColor','w');
view(45,30);
title('3D Multi-Drone Mission with Conflict Detection','Color','w','FontSize',14);


% Plot static trajectories for reference
plot3(x_main,y_main,z_main,'b--','LineWidth',1.5);
colors = [1 0 0; 0 0.6 1; 0 1 0]; % Drone 1 red, Drone 2 sky-blue, Drone 3 green
for i=1:numel(otherDrones)
plot3(otherDrones(i).x,otherDrones(i).y,otherDrones(i).z,'--
','Color',colors(i,:),'LineWidth',1.5);
end


% Drone markers
hMain = plot3(NaN,NaN,NaN,'bo','MarkerFaceColor','b','MarkerSize',10);
hOthers = gobjects(1,numel(otherDrones));
for i = 1:numel(otherDrones)
hOthers(i) = plot3(NaN,NaN,NaN,'o','MarkerFaceColor',colors(i,:),'MarkerSize',8);
end


% Conflict marker (yellow)
hConflict = plot3(NaN,NaN,NaN,'yo','MarkerFaceColor','y','MarkerSize',12);


%% ======== Real-Time Animation with Randomized Conflicts ========
rng('shuffle'); % random seed
for k = 1:length(t_main)
% Update primary drone
set(hMain,'XData',x_main(k),'YData',y_main(k),'ZData',z_main(k));
conflictFlag = false;
conflictPos = [];
conflictDroneIDs = [];
% Update other drones
for i=1:numel(otherDrones)
set(hOthers(i),'XData',otherDrones(i).x(k),...
```

```
'YData',otherDrones(i).y(k),...
'ZData',otherDrones(i).z(k));
end
% Randomized conflict generation
if rand < 0.05 % 5% chance per timestep
conflictFlag = true;
conflictPos = [x_main(k)+randn*2, y_main(k)+randn*2, z_main(k)+randn*2];
conflictDroneIDs = randi([1,numel(otherDrones)],1); % random drone causing conflict
fprintf("⚠ Conflict at t=%.2f sec with Drone %d at (%.2f, %.2f, %.2f)\n", ...
t_main(k), conflictDroneIDs, conflictPos(1), conflictPos(2), conflictPos(3));
end
% Update conflict marker
if conflictFlag
set(hConflict,'XData',conflictPos(1),'YData',conflictPos(2),'ZData',conflictPos(3));
else
set(hConflict,'XData',NaN,'YData',NaN,'ZData',NaN);
end
drawnow;
end


%% =========================
% Function Definitions
%% =========================


function [status, conflictDetails] =
queryMission(x_main,y_main,z_main,t_main,otherDrones,safetyBuffer)
% Checks for spatial-temporal conflicts before simulation
conflicts = [];
for i = 1:numel(otherDrones)
for k = 1:length(t_main)
dx = x_main(k) - otherDrones(i).x(k);
dy = y_main(k) - otherDrones(i).y(k);
dz = z_main(k) - otherDrones(i).z(k);
dist = sqrt(dx^2 + dy^2 + dz^2);
if dist < safetyBuffer
conflicts(end+1,:) = [otherDrones(i).x(k), otherDrones(i).y(k), otherDrones(i).z(k),
t_main(k), i]; %#ok<AGROW>
end
end
end
if isempty(conflicts)
status = "Clear";
conflictDetails = [];
else
status = "Conflict Detected";
conflictDetails = array2table(conflicts, 'VariableNames',{'X','Y','Z','Time','DroneID'});
end
End
```

OUTPUT:

The simulation video is attached with the folder.