Emerson Hall

Ramana M. Pidaparti

1 September 2025

Simulation Environment Setup

Introduction

The objective of Week 3 was to implement a simulation environment for unmanned aerial vehicle (UAV) trajectory using MATLAB R2024b and the UAV Toolbox. Simulation provides a safe and repeatable method to evaluate navigation strategies before deploying them in physical hardware. For this week, a simplified environment was developed consisting of a ground plane, two static obstacles and a quadrotor platform executing a waypoint mission. This simulation environment establishes the foundation for evaluating baseline and machine learning based trajectory optimization techniques.

Methods

The simulation was constructed within the uavScenario (Link) framework in MATLAB. The environment was defined with a 20 Hz update rate and a stop time of 60 seconds. A surface mesh represented the ground plan, while two polygonal prisms modeled obstacles (a thin wall and a block resembling a building).

A quadrotor platform was introduced using the uavPlatform function and visualized with a scaled “quadrotor” mesh. The UAV’s motion was governed by a kinematic waypoint tracker that advanced the vehicle toward predefined waypoints. This tracker calculated direction vectors toward each waypoint, applied a velocity constraint of 1.2 m/s, and updated the vehicle’s state using the move command. Visualization was achieved through show3D, which rendered both the environment and the UAV’s trajectory in real time.

The waypoint mission required the UAV to ascend, navigate along the wall, pass through an opening, go around the building and descend to a designated final position. A tolerance of 0.35 m was used to determine waypoint completion.

Result

The UAV successful executed the mission plan within the simulated environment. The trajectory demonstrated proper avoidance of the wall and building which confirms that the waypoint tracking algorithm enabled effective navigation around obstacles. The visualization displayed the UAV following the planned path while leaving a persistent trail for trajectory validation. At the conclusion of the mission, the simulation displayed the message “Mission Complete” to verify that the UAV reached its final way point.

A figure of the simulated environment was captured, illustrating the UAV, obstacles, and trajectory path. This figure is included below as evidence of the successful demonstration.

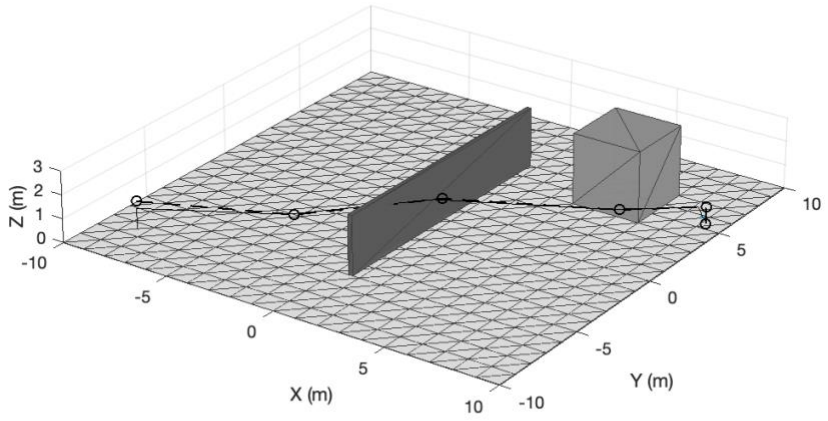

Figure 1: UAV navigating a waypoint path around a wall and building in MATLAB (R2024b)

Discussion

The week 3 implementation demonstrates feasibility of MATLAB's UAV Toolbox for rapid prototyping of UAV trajectory scenarios. The simulation confirmed that a kinematic tracker can handle waypoint guidance in simple environments with static obstacles. Although the approach is effective for demonstration, it lacks dynamic adaptability such as real-time replanning or collision avoidance against moving entities.

Future work can expand this baseline by incorporating path planning algorithms such as A* or RRT for more complex environment. Additionally, the simulation can be extended to include external disturbances sensor noise or multi-UAV coordination. Ultimately, this environment provides a steppingstone for applying reinforcement learning approaches to UAV trajectory optimization in later project phases.

Conclusion

A functional UAV simulation environment was created in MATLAB during week 3. The platform successfully executed a waypoint mission in a scenario containing two obstacles. This step fulfills the milestone of week 3 of establishing an environment for testing UAV trajectory algorithms. The work paves the way for more advanced planning and machine learning methods in the upcoming stages of the research project.

Appendix

A: MATLAB Code

```
%% Week 3 Demo (R2024b): UAV waypoint mission in uavScenario

clear; clc; close all;

%% 1) Build scenario
scene = uavScenario( ...
    "UpdateRate", 20, ...     % Hz
    "StopTime",   60,  ...    % s
```

```
    "ReferenceLocation", [0 0 0]);  % ENU

% Ground (20 m x 20 m) as SURFACE
[xg, yg] = meshgrid(-10:1:10, -10:1:10);
zg = zeros(size(xg));
addMesh(scene, "surface", {xg, yg, zg}, [0.85 0.85 0.85]);  % light gray

% Thin wall as POLYGON prism  (corners Nx2, height [zmin zmax])
wallCorners = [ 0.5 -6;
                0.7 -6;
                0.7  6;
                0.5  6 ];
addMesh(scene, "polygon", {wallCorners, [0 2.5]}, [0.35 0.35 0.35]);

% Square block as POLYGON prism
bldgCorners = [ 4 4;
                7 4;
                7 7;
                4 7 ];
addMesh(scene, "polygon", {bldgCorners, [0 3.0]}, [0.55 0.55 0.55]);

%% 2) Platform and mesh
uav = uavPlatform("quad", scene, "ReferenceFrame", "ENU");

% Mesh: name, {scale}, color, 4x4 offset (identity)
updateMesh(uav, "quadrotor", {0.25}, [0 0.45 0.74], eye(4));

% Helper for quaternion from yaw (about z)
yaw2quat = @(psi)[cos(psi/2), 0, 0, sin(psi/2)];

% Initial state
pos   = [-8, -8, 0.3];
yaw   = 0;
vel   = [0 0 0];
acc   = [0 0 0];
angv  = [0 0 0];
quat  = yaw2quat(yaw);

% Seed platform pose
move(uav, [pos, vel, acc, quat, angv]);

%% 3) Waypoints (x,y,z) in ENU
wps = [ ...
   -8  -8  1.5;   % ascend
   -2  -6  1.8;   % along the wall
    2   -2  2.0;  % through gap
```

```
    8    1  2.0;  % past block
   10    4  1.5;  % approach
   10    4  0.8]; % descend

dt       = 1 / scene.UpdateRate;
maxSpeed = 1.2;      % m/s
tolReach = 0.35;     % m
idx      = 1;

%% 4) Prepare + visualize
setup(scene);
fig = figure("Name","UAV Mission (R2024b)");
ax  = show3D(scene);                    % create axes once
view(35,25); camproj('perspective'); grid(ax,'on'); axis(ax,'equal');
xlabel(ax,'X (m)'); ylabel(ax,'Y (m)'); zlabel(ax,'Z (m)'); hold(ax,'on');
plot3(ax, wps(:,1), wps(:,2), wps(:,3), 'k--o', 'LineWidth', 1);
trail = animatedline('Parent', ax, 'LineStyle','-', 'LineWidth',1);

%% 5) Kinematic waypoint tracker
while scene.CurrentTime < scene.StopTime
    wp = wps(idx,:);
    d  = norm(pos - wp);

    if d < tolReach
        idx = min(idx + 1, size(wps,1));
        wp  = wps(idx,:);
        d   = norm(pos - wp);
        if idx == size(wps,1) && d < tolReach
            break;  % final waypoint reached
        end
    end

    % velocity toward waypoint (clamped)
    dir = wp - pos;
    if norm(dir) > 1e-6
        vel = maxSpeed * dir / max(norm(dir), 1e-6);
    else
        vel = [0 0 0];
    end

    % integrate simple kinematics
    pos = pos + vel * dt;

    % fixed yaw (0); keep it simple
    yaw  = 0;
    quat = yaw2quat(yaw);
```

```
    % Move platform: [x y z vx vy vz ax ay az qw qx qy qz wx wy wz]
    motion = [pos, vel, [0 0 0], quat, [0 0 0]];
    move(uav, motion);

    % step simulation + fast redraw on same axes
    advance(scene);
    show3D(scene, "Parent", ax, "FastUpdate", true);

    % trail
    addpoints(trail, pos(1), pos(2), pos(3));
end

disp('Mission complete.');
```