Universit´e d’Evry-Val-d’Essonne, UFR Sciences et Technologies

Master2 E3A: Smart Aerospace & Autonomous Systems

Mission Coordination TP1

By: Abel Mebratu Yehuala ID = 20245853

Melkamu Amare ID = 20245847

Submitted to: Prof. Sofiane AHMED-ALI

Abstract

This report investigates the dynamics and control of an Unmanned Ground Vehicle (UGV) system equipped with two independently controlled motors. We derived and analyzed the state-space representation of the UGV, a control strategy, including a PD and P controller, are implemented to regulate the system's outputs. Simulations are performed in MATLAB/Simulink to examine the open-loop and closed-loop behaviors, ensuring precise tracking of reference values for orientation (θ) and velocity. The x and y trajectory of UGV were done in Simulink. Additionally, the x and y trajectory of the UGV is simulated and visualized in Simulink.

Introduction

Unmanned Ground Vehicles (UGVs) are complex systems that require a thorough understanding of their dynamics, state-space representation, and control strategies for effective operation. The dynamics of a UGV are typically defined by five state variables: x and y for position in 2D space, θ for orientation angle (heading), and v1 and v2 for the velocities of the left and right wheels respectively. The kinematic model of a UGV describes its motion without considering forces.

A control strategy applied to UGVs. The Proportional-Derivative (PD) controller and the Proportional controller designed for both rotational and linear movement.

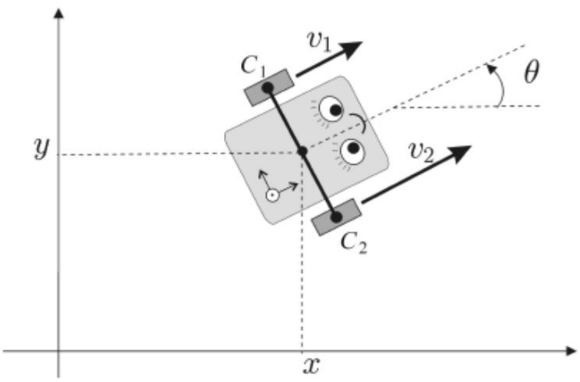


Figure 1: UGV model

UGV dynamics defined by its state variables

$$\dot{x} = \frac{1}{2}(v_1 + v_2)\cos(\theta)$$
$$\dot{y} = \frac{1}{2}(v_1 + v_2)\sin(\theta)$$
$$\dot{\theta} = \frac{1}{l}(v_1 - v_2)$$
$$\dot{v}_1 = Ru_1$$
$$\dot{v}_2 = Ru_2$$

The state-space representation of the UGV is a nonlinear system. This is because the kinematic equations for x, y, and theta include trigonometric terms (cos (θ) and sin (θ)), which are **nonlinear functions** of the state variables. However, the dynamic equations for v1 and v2 are linear with respect to the control inputs u1 and u2.

Materials and Methods

Materials

- Software: MATLAB R2023b and R2024b and Simulink
- UGV model.

Methods

Step 1:

Define the outputs y_1 and y_2 as follows:

$$y_1 = \theta$$

$$y2 = \frac{v1 + v2}{2}$$

Step 2:

Compute the second derivative of the theta and first derivative of linear velocity

$$\ddot{y}_1 = \frac{R}{l}(u_1 - u_2)$$

$$\dot{y}_2 = \frac{1}{2}(\dot{v}_1 + \dot{v}_2) = \frac{1}{2}(Ru_1 + Ru_2) = \frac{R}{2}(u_1 + u_2)$$

Step 3:

Define the control law for state feedback:

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = A_y^{-1} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

The above equation shows a linear relationship between the vector of output derivatives and the input vector through the inverse of a matrix

Step 4:

Controller design based on the equation

Inputs: Errors and reference derivatives

$$e_1 = y_{ref} - y_1;$$

$$\dot{e} = \dot{y_{ref}} - \dot{y_1}$$

$$e_2 = y_2 - y_{ref}$$

y1_ref_ddot and y2_ref_dot are computed based on references.

PD Controller for y1

$$u_1 = \frac{l}{R}\left(K_{p1}e_1 + K_{d1}\dot{e}_1 + y_{1\_ref\_ddot}\right)$$

P Controller for y2

$$u_2 = \frac{2}{R}\left(K_{p2}e_2 + y_{2\_ref\_dot}\right)$$

Open-Loop Analysis

Physical Significance of y_2

$$y2 = \frac{v1 + v2}{2}$$

- y2 represents the average forward velocity of the UGV, a critical parameter for ensuring consistent motion.

To achieve pure rotational motion, the control inputs for the UGV are defined as:

$$u1 = -u2$$

The above equation indicates that the control input for the left wheel (u1) should be equal in magnitude but opposite in direction to the control input for the right wheel (u2). This ensures that there is a pure rotational movement around the center of gravity (CG) of the UGV.

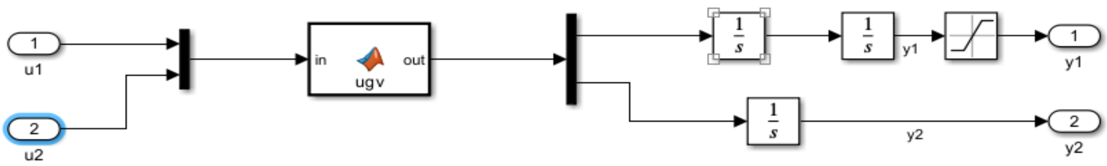


Figure 2: Open-Loop of UGV

Closed-Loop Control of UGV

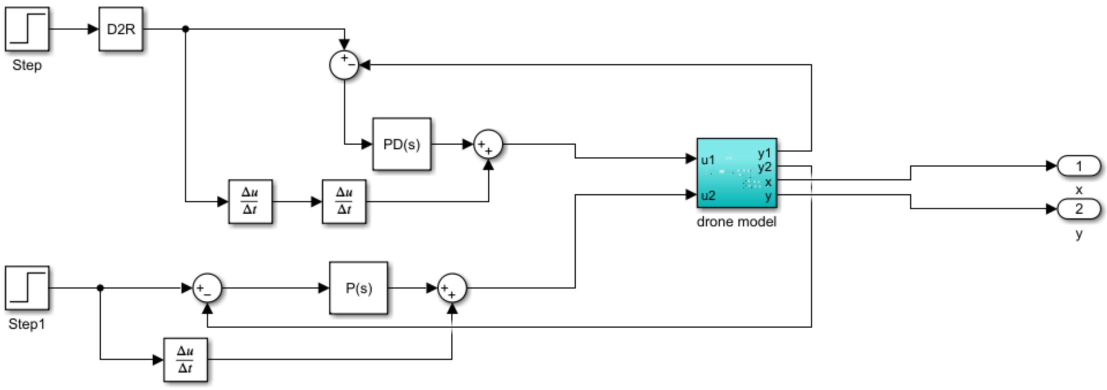


Figure 3: Closed-Loop of UGV

Result and Analysis

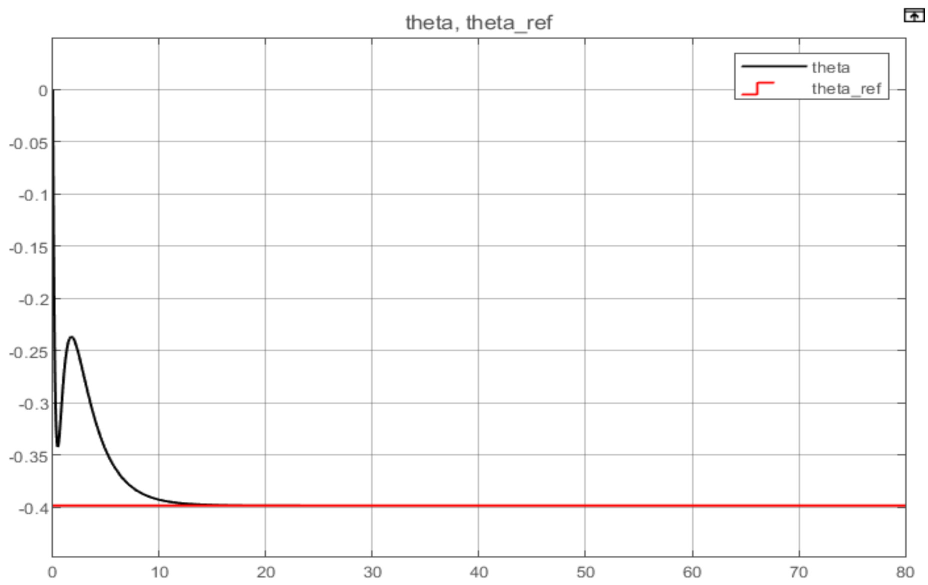


Figure 4a: Time response of theta and its reference

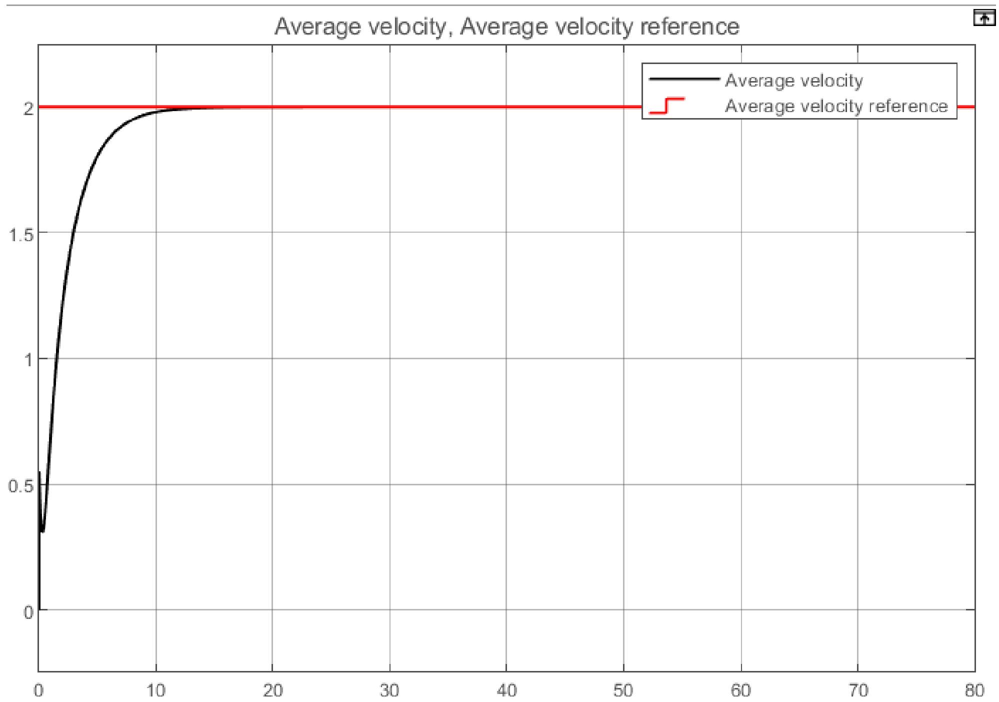


Figure 4b: Time response of average velocity and its reference

As shown in the figure 4b, initially, the UGV's velocity starts at 0 and quickly rises towards the reference velocity. The system exhibits a slight **overshoot** during the initial transient response before settling. Once the transient effects diminish, the UGV maintains a steady average velocity equal to the reference velocity. The controller effectively regulates the UGV's velocity, ensuring that it tracks the desired reference speed with minimal error.

As shown in figure 4a, the initial orientation angle (θ) begins at approximately **0 radians** and undergoes a transient response before settling at the desired reference value of **-0.4 radians**. The system shows a slight **oscillation** at the start, where the angle temporarily moves in the opposite direction before stabilizing. After a few seconds, the orientation settles closely to the reference value with minimal error and remains constant over time. The controller efficiently adjusts the UGV’s orientation to track the desired reference angle

Mission coordination of the UGV

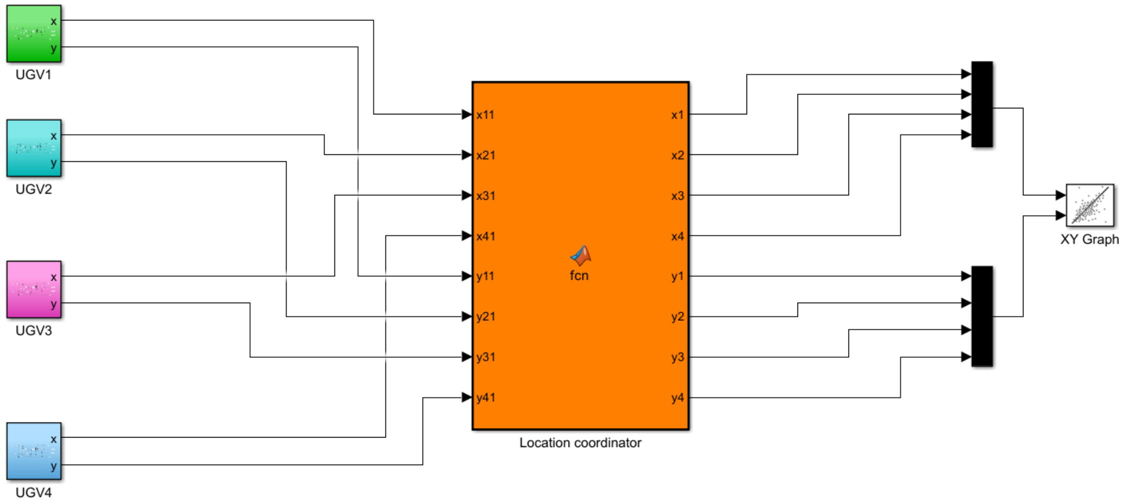
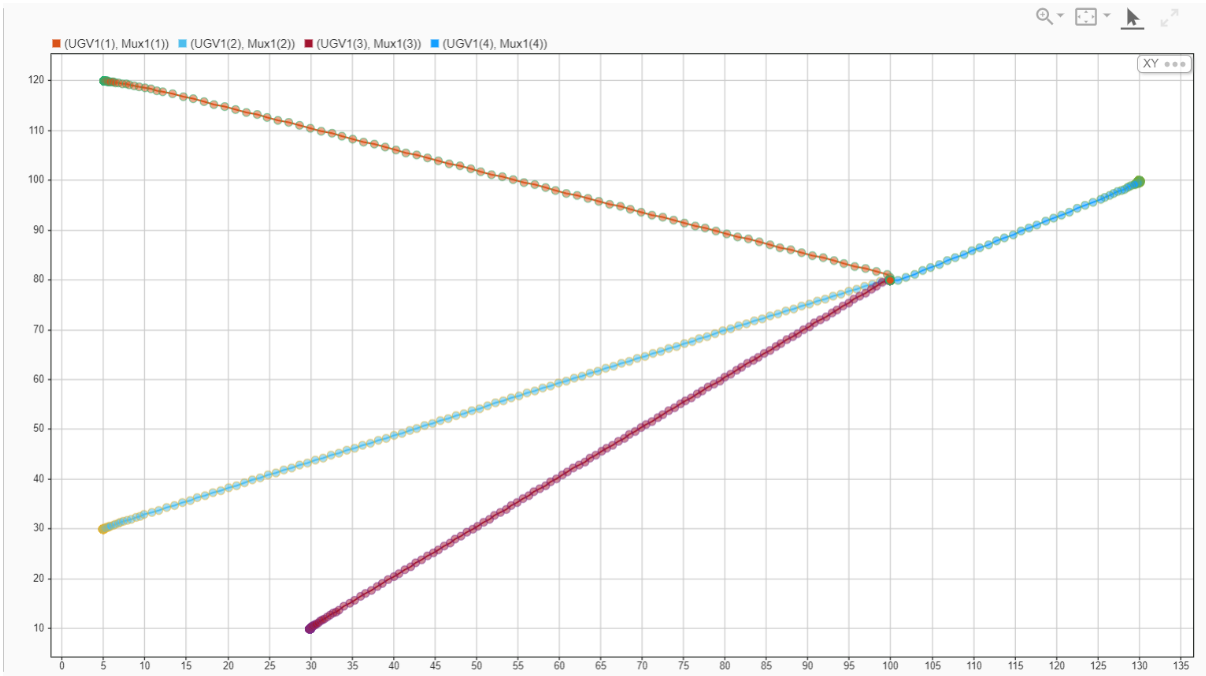


Figure 5: Simulink model of the four Unmanned Ground Vehicles



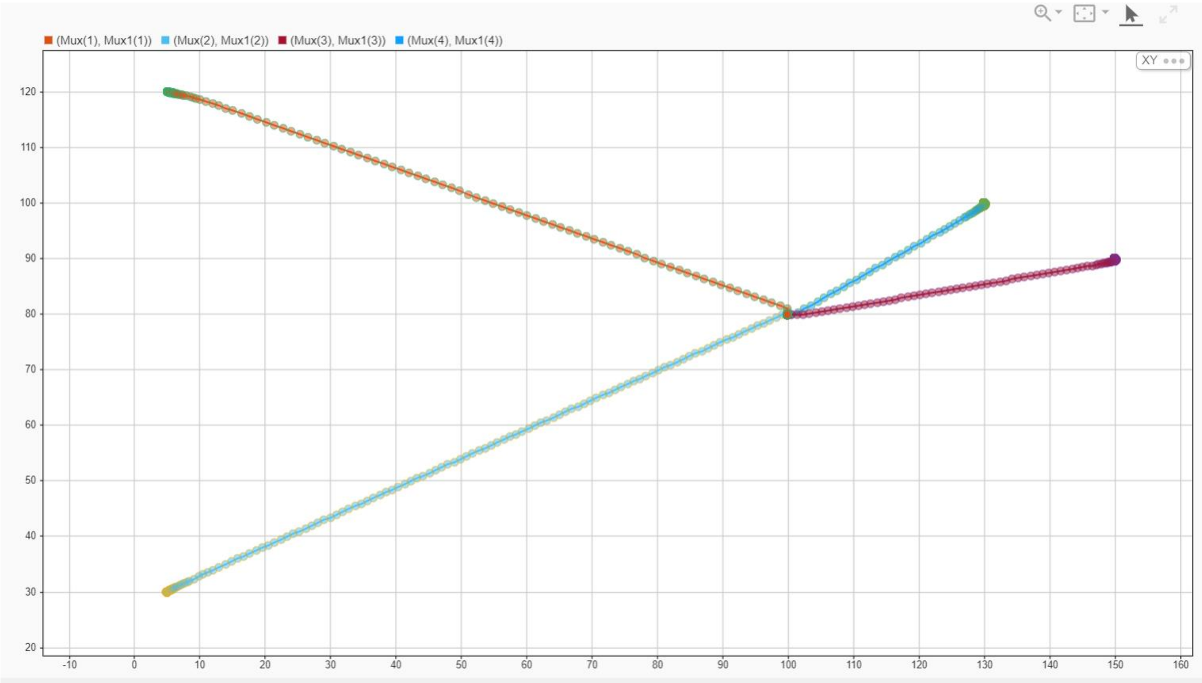


Figure 6: The trajectory of the four Unmanned Ground Vehicles

The above figure illustrates the trajectory of four Unmanned Ground Vehicles (UGVs), each starting from **different initial positions**. Our main goal was to converge all the UGVs at the **same final point** using appropriate control strategies. The strategies we used ensured the convergences while maintaining proper trajectory.

Control Strategy

Velocity and Orientation Control:

Each UGV's motion is regulated by controlling its **average velocity** and orientation using proportional (P) and proportional-derivative (PD) controllers. The average velocity ensures that the UGV moves forward at a controlled rate, while the orientation control adjusts the UGV's heading to align with the path leading to the target point.

The Control strategy we used for mission coordination task of the UGV:

Initial Conditions:

- x10, y10, x20, y20, x30, y30, x40, y40 are the starting positions of the four UGVs.
- The UGVs move toward a common meeting point defined by x_lim and y_lim.

Controlled Inputs for the location coordinator block:

- x11, x21, x31, x41: Updated x-positions for each UGV.
- y11, y21, y31, y41: Updated y-positions for each UGV.

Position restriction:

For the x-axis:

- If a UGV moves past x_lim, its x-position is clamped to x_lim if the current position (x11) crosses the boundary in either direction relative to the previous position (x10).
- Similar process for the y-axis, ensuring positions do not exceed y_lim.

Meeting Point:

- We defined (x_lim, y_lim), where all UGVs will converge.
- The function ensures all UGVs are restricted to this final position once they reach or cross it.

Conclusion

In this report, the control and coordination of an **Unmanned Ground Vehicle (UGV)** system with two independent controllers were analyzed. The system dynamics were modeled, and **PD (Proportional-Derivative)** and **P (Proportional)** control strategies were implemented to regulate the UGV's orientation (θ) and average velocity (v), respectively. Simulation results demonstrated that the PD controller ensured angular stability, while the P controller achieved precise velocity tracking.

The UGV's motion was simulated in MATLAB/Simulink for both open-loop and closed-loop behaviors. A position-limiting control strategy was applied to coordinate multiple UGVs, enabling them to start from different initial positions and converge to a common target point (xlim, ylimx). The position-limiting function effectively prevented overshoot, ensuring stable and controlled convergence, which highlights the control system's effectiveness in managing multi-UGV tasks.

References

1. Sofiane AHMED-ALI. Mission coordination for multi-robot's lecture note
2. Wiley.Ogata, K. (2010). Modern control engineering (5th ed.)

Appendices

```
function [x1,x2,x3,x4,y1,y2,y3,y4] = fcn(x11,x21,x31,x41,y11,y21,y31,y41, x10, y10, x20, y20, x30, y30, x40, y40, xlim, ylim)
% x-axis limit evaluation
x1 = x11;
if x11 > xlim && x10 < xlim
x1 = xlim;
elseif x11 < xlim && x10 > xlim
x1 = xlim;
end
x2 = x21;
if x21 > xlim && x20 < xlim
x2 = xlim;
elseif x21 < xlim && x20 > xlim
x2 = xlim;
end
x3 = x31;
if x31 > xlim && x30 < xlim
x3 = xlim;
elseif x31 < xlim && x30 > xlim
x3 = xlim;
end
x4 = x41;
```

```
if x41 > xlim && x40 < xlim
x4 = xlim;
elseif x41 < xlim && x40 > xlim
x4 = xlim;
end
% y-axis limit evaluation
y1 = y11;
if y11 > ylim && y10 < ylim
y1 = ylim;
elseif y11 < ylim && y10 > ylim
y1 = ylim;
end
y2 = y21;
if y21 > ylim && y20 < ylim
y2 = ylim;
elseif y21 < ylim && y20 > ylim
y2 = ylim;
end
y3 = y31;
if y31 > ylim && y30 < ylim
y3 = ylim;
elseif y31 < ylim && y30 > ylim
y3 = ylim;
end
y4 = y41;
if y41 > ylim && y40 < ylim
y4 = ylim;
elseif y41 < ylim && y40 > ylim
y4 = ylim;
end
end
```