INTERNATIONAL JOURNAL OF AUTOMOTIVE AND MECHANICAL ENGINEERING (IJAME)
ISSN: 2229-8649 e-ISSN: 2180-1606
VOL. XX, ISSUE XX, XXX – XXX
DOI: https://doi.org/10.15282/ijame.v#i#.###

ORIGINAL ARTICLE

Interactive Simulation Framework for Analysing Tracked Mobile Robots in Real-Time

Low Chow Yeh[1]*, Vimal Rau Aparow[1], Hermawan Nugroho[1*]

[1]Electrical and Electronic Engineering Department, University of Nottingham Malaysia, 43500 Semenyih, Malaysia

ABSTRACT – Tracked mobile robots play a crucial role in navigating complex terrains for urban search and rescue and military applications, yet verifying their performance in diverse environments remains a challenge. This study introduces an interactive simulation framework using a Hardware-in-the-Loop (HIL) platform to analyse real-time performance, focusing on critical capabilities such as skid-steering, slope handling, and obstacle climbing. By integrating detailed kinematic models with a virtual environment powered by Unreal Engine, the framework delivers precise simulations that closely replicate real-world scenarios. Validation tests revealed a maximum position error of ±8.5838 cm, with mean squared errors of 0.1854 m for the x-coordinate, 0.1486 m for the y-coordinate, and 0.801 radians for the yaw angle in straight-line navigation. Despite higher errors in complex manoeuvers, the results demonstrate the framework's effectiveness in bridging simulation and real-world performance, providing a reliable tool for the design and testing of mobile robots in challenging environments.

ARTICLE HISTORY
Received: xxxx
Revised: xxxx
Accepted: xxxx
Published: xxxx

KEYWORDS
Hardware-in-the-loop
Skid-steering
Tracked mobile robot
Unreal engine
Real-time simulation

Pre-Proof Copy

1.0 INTRODUCTION

Robot locomotion refers to various methods for the movement of a robot [1]. Compared to other types, wheeled robots are popular choices. This is due to their control simplicity. Steering in real-world environments often requires overcoming obstacles and various types of terrains. Most wheeled robots, unfortunately struggle with these irregular terrains [2]. This constraint emphasises the importance of developing "all-terrain robots" that can run and handle various hurdles. Such robots would improve the abilities and application of robots.

Tracked mobile robots are recognised for their ability to steer through terrains. As such, these types of robots can be applied and play an important role in urban search and rescue missions and military operations. Different from wheeled robots, tracked robots offer stability and adaptability. Their design and testing, however face difficulties in precisely replicating real-world conditions.

In the development and design of a robot, simulation of the design and its testing are critical to confirm that the robotic software meets the specified requirements [3], [4], [5], [6]. The testing should enable the testing of the robot's subsystems, such as its control systems, localisation, and object detection, in a virtual environment before the subsystems are adopted in real-world applications. Generally, computer simulation is a favoured method in the early stages of robot development due to its time and cost efficiency. This is especially important in the design of experimental robots for certain tasks/aims. For instance, here the authors develop an integrated route and path planning strategy for a skid steering robot, which will be used to harvest in agricultural environments with terrain constraints [7]. Liang et al. designed a model-based coordinated trajectory tracking control approach for a four-wheel. The robot uses a skid-steer control system equipped with a timing-belt servo system [8]. Moreno et al. developed a linearisation-based trajectory tracking controller of input and output for skid steering robots [9]. Transitioning from virtual to real environments often necessitates the substitution of simulation models with actual robots and the adaptation of control logic, potentially leading to new software issues that emerge in physical settings. Hardware-in-the-Loop (HIL) simulation addresses this challenge by integrating real robot hardware with simulated environments, enhancing the reliability of the robot's performance to mirror the simulation, and reducing discrepancies between simulated and actual testing.

Numerous robot simulators exist to study mobile robot behaviour. MATLAB Simulink, for instance, is popular for robot system development [10], [11], [12], [13], [14]. Tran D et al. use MATLAB Simulink simulation platform to test and evaluate the stability of a cable-driven hyper-redundant robot in a teleoperation system. The simulation is used to ensure the safety of the system before deployment [15]. Moreno et al. also use the MATLAB Simulink simulation platform to test the deployment of their developed, which presents a real-time control system for a two-degree-of-freedom (2DOF) robot. The simulation is aimed for FPGA deployment and is applied to validate the system's robustness against noise and disturbances [16]. Tristano M et al. design a vehicle stability control system that uses individual wheel torque. MATLAB Simulink simulation platform is used to validate and progress the model to a hardware-in-the-loop (HIL) setup incorporating an Electronic Control Unit (ECU [17]. Dosoftei C et al. develop an omnidirectional mobile robot (OMR)

*CORRESPONDING AUTHOR | H. Nugroho | ✉ hermawan.nugroho@nottingham.edu.my

© The Authors 2019. Published by Penerbit UMP. This is an open access article under the CC BY license.

in dynamic logistics environments. They combine MATLAB-Simulink with the Robot Operating System (ROS) to employ and test the algorithms [18]. For Hardware-in-the-Loop (HIL) simulations, MATLAB Simulink has challenges in real-time execution due to hardware integration issues arising from compatibility constraints, requiring additional toolboxes. Scalability matters, and high licensing costs make the framework less practical. Other simulations, such as the Virtual Robot Experimentation Platform (V-REP), offer a 3D environment with an integrated development environment (IDE) [19], [20], [21]. Gazebo excels at realistic sensor feedback and object interactions but is often used for drone research [22], [23], [24], [25]. Specific simulators like Truck Maker cater to specialised applications [26], [27], [28]. Many of these simulators rely on predefined kinematic models, which restrict customisation and adaptability to different robot designs, especially for skid-steering tracked robots. This is where our approach diverges.

A common limitation among existing robot simulators is their inability to accommodate arbitrary robot designs, particularly those with diverse and specialised locomotion types. Many traditional simulators come with predefined kinematic models that are fixed and not easily customisable, which restricts their flexibility in simulating different types of robots, especially tracked mobile robots. This rigidity hinders the accurate modelling and testing of robots with unique or unconventional locomotion mechanisms, ultimately limiting the usefulness of these simulators for researchers and engineers working on innovative robotics applications. Moreover, much of the existing research and development has focused on simulating wheeled or legged robots, with significantly less attention given to tracked mobile robots. These robots, essential for environments requiring stability and traction over rough terrain, present unique challenges that are not adequately addressed by simulators designed for other locomotion types. The lack of simulation tools for the development of tracked robots makes the engineers to compromise. As a result, the developed robots may have discrepancies between their simulated result and real-world performance.

To address such challenge, in this project, we develop a Hardware-in-the-Loop (HIL) simulation platform, which is specifically adapted for tracked mobile robots using Unreal Engine. Different from traditional simulators, which are limited by fixed kinematic models, Unreal Engine enables custom robot models to be integrated with its highly flexible environment. The flexibility of the framework will allow us to customise the robot's kinematics and 3D physical modelling. This customisation will enable precise simulation of the distinctive movement of tracked robots. Unreal Engine’s broad set of tools includes an effective graphics engine, a flexible programming language, and a vigorous physics engine. Such tools will allow the simulation of real-world physics, such as gravity, collisions, and friction, with high fidelity. This makes it exceptionally well-suited for simulating robots in complicated settings where traditional physical testing might be impractical. By leveraging the technology of this game engine, this project not only overcomes the limits of existing simulators but can also offer a strong platform for the design, testing, and validation of tracked mobile robots in situations that carefully reproduce real-world challenges.

There are various simulators for mobile robot testing, such as MATLAB Simulink, Gazebo, and V-REP. These programs, however, come with limitations, especially when simulating complicated and complex situations or customised robots such as robots with unconventional locomotion mechanisms. Many of these simulators use predefined kinematic models. Such settings limit the customisation and adaptability of robot designs, especially for skid-steering tracked robots. This is where our approach varies. The uniqueness of our approach is that our approach offers complete flexibility. It enables to model unconventional locomotion, as well as the integration of real-time Hardware-in-the-Loop (HIL) simulations. The HIIL enables immediate feedback between virtual environments and physical hardware. This approach shows that our simulations not only encapsulate the design of tracked robots in varied situations but also lessen discrepancies between simulated and real-world performance. This indicates that the approach is versatile and can be used to test tracked mobile robots accurately.

2.0 METHODS AND MATERIAL

The Hardware-in-the-Loop (HIL) system lets users examine the real-time integration of virtual environments and physical systems and/or the subsystems of the robot. It allows users to evaluate the system and/or the subsystem's performance under real-world conditions without the demand of a full physical setup.

In this project, we develop a novel adaptation of the kinematic approach to be adopted for tracked mobile robots to improve motion control and pose estimation. The approach is expected to be able to calculate the motion of tracked mobile robots using only track velocities. Such calculation is tricky due to complex dynamics, including slippage and track-ground interactions. In the first subsection, we explore the kinematics of skid-steering and obstacle-climbing, laying the groundwork for the simulation model. Next, we discuss the development of the simulation environment using Unreal Engine.

2.1 Kinematics Model as Robot Simulation Model

Tracked mobile robots use skid-steering for movement, which means each side of the robot can move independently, allowing it to turn by adjusting the speed difference between the left and right tracks. However, skid-steering can cause slippage, particularly during turns or on uneven surfaces. To address this, we implemented a kinematic model that includes slip compensation. The kinematic model utilises the speed of the robot and its track conditions to predict the position and orientation of the robot during movement, with a focus on controlling the robot in real-world environments.

Pre-Proof Copy

Skid-steering locomotion is a locomotion setting of tracked vehicles such as tanks and bulldozers. The locomotion lets the vehicle independently control the speed and direction of the tracks that control the vehicle by adjusting the speed difference between the tracks [29]. Skid-steering, however, can be slipped during turns. To address this challenge and improve the prediction of its motion, a slip-compensated odometry model is usually adopted in the kinematic models. In this project, we utilise the kinematics of a tracked car with skid-steering movement defined by Rigatos [30], [31], [32].

In skid-steering, each track operates independently to control the movement of the tracked mobile robots. This mechanism, however, instigates slippage, notably during turns or on uneven surfaces. To handle this issue, a slip compensation is implemented and incorporated into the kinematic model. The developed kinematic model approximates the position and orientation of the tracked robot based on the velocity and surface conditions, which is then used for motion prediction. In the kinematic model, we adopt formulations to express forward velocity, lateral movement, and yaw angle changes, factoring in slip ratios and track geometry. These formulations facilitate effective simulation of the robot and its exchanges with obstacles, such as climbing steps or navigating slopes. These formulations oversee the forward motion, turning, and interaction with obstacles for skid-steering locomotion, as follows:

$$V_x = \frac{v_r(1-\alpha_r) + v_l(1-\alpha_l)}{2} \tag{1}$$

The formula determines the forward velocity of the robot (V_x) calculated based on the velocities of the left and right tracks (v_r and v_l), modified by their respective slip ratios (α_r and α_l).

$$V_y = V_x \cdot \tan\beta \tag{2}$$

The lateral velocity (V_y) is calculated and determined based on the forward velocity (V_x) and the slip angle (β). This accounts for any sideways movement caused by track slippage.

$$\dot{x} = V_x \cos\theta - V_y \sin\theta \tag{3}$$

$$\dot{y} = V_x \sin\theta + V_y \cos\theta \tag{4}$$

$$\dot{\theta} = \frac{v_r(1-\alpha_r) - v_l(1-\alpha_l)}{L} \tag{5}$$

The change in yaw angle ($\dot{\theta}$) is determined by the difference in track velocities and their slip ratios, divided by the equivalent wheelbase (L). This equation describes how the robot turns based on the difference in speeds between the two tracks. If the car runs without experiencing longitudinal slippage, it indicates that its ground velocity matches the input velocity and a slip ratio of zero [33]. The equivalent wheelbase of the robot is represented by the parameter L. The illustration of the kinematics model of the tracked car can be seen in Fig. 1.

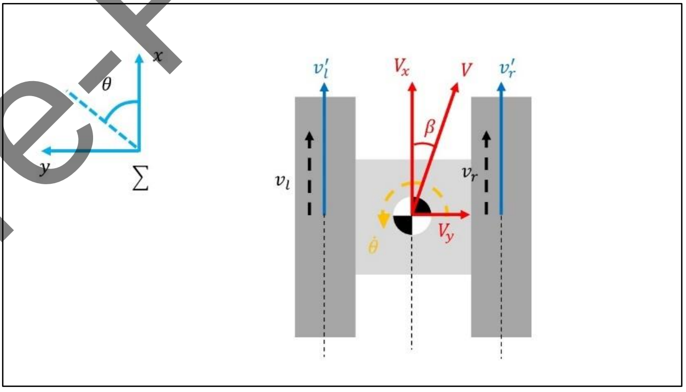


Fig. 1 Kinematics model of tracked car

Obstacle climbing is a critical capability for tracked locomotion robots. It allows these robots to navigate diverse urban structures, including stairs, small steps, and road bumps. When tackling an obstacle—such as a step—the process involves two primary stages: 1) Step Climbing: The robot ascends the obstacle, and 2) Step Crossing: The robot traverses the obstacle horizontally [34].

Step climbing refers to the ability of a tracked mobile robot to ascend vertical obstacles with distinct steps. During this manoeuvre, the robot utilises the adhesive force of its tracks to counteract gravity and propel itself upwards. Fig. 2 illustrates the tracked car leverages its tracks for step climbing. When the robot reaches the vertical step (riser), the tracks generate a driving force that causes the body to rotate counterclockwise, lifting the front section in preparation for climbing the step.

Pre-Proof Copy

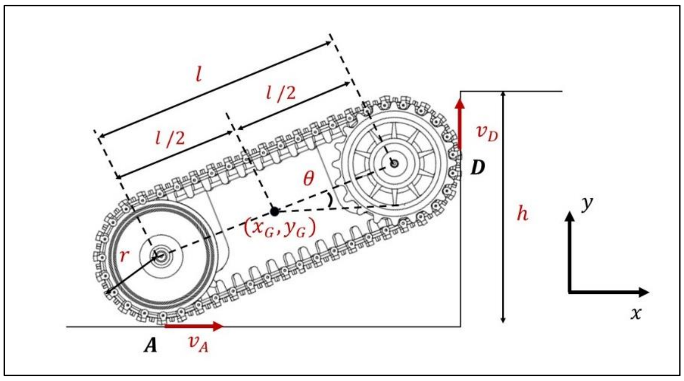


Fig. 2 Kinematics of Step Climbing

For the model formulation, the origin of the right-handed Cartesian coordinate system (x_G,y_G) is aligned and defined at the centre of the tracked car [35]. The design of the track model prioritises minimising slippage during the climbing motion. To achieve this, the model assumes no track sliding or slippage occurs at points A and D (refer to Figure 2 for reference). With this assumption, the robot's displacement from point D can be estimated using the following equation :

$$\Delta y = \int v_{wheel}\; dt \tag{6}$$

If the displacement Δy is less than the obstacle's height minus the wheel radius (h-r), we define the robot's rotation angle and speed. Should Δy exceed this value, it indicates the wheel has reached the step's edge. The angle α, initially 0 degrees, measures the tilt against the horizontal, increasing to 90 degrees as the wheel ascends the step. The climb completes when α surpasses 90 degrees, transitioning to step crossing as shown in Fig. 3. The robot's rotation, speed, and mass centre are determined by specific formulas. After climbing, the robot crosses the step, as shown in Figure 4.

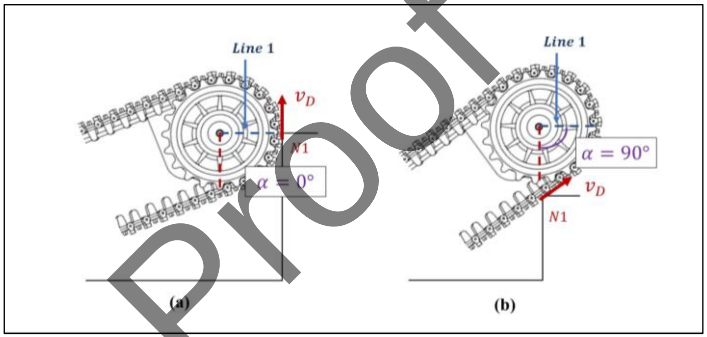


Fig. 3 Different value of α; (a) α = 0° (b) α = 90°

. The length L_D signifies the distance from point A to the step's edge, which decreases gradually. When L_D becomes less than zero, the step-crossing process concludes. The equations below are used to derive the robot's body rotation angle, angular velocity, centre of mass position, and velocity.

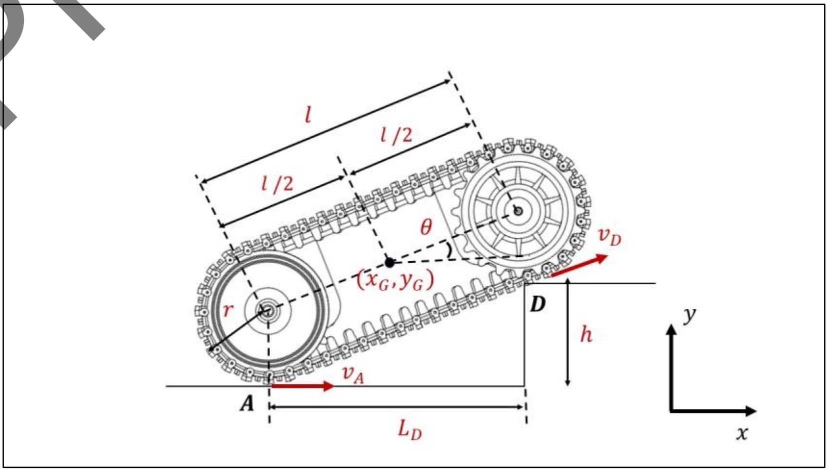


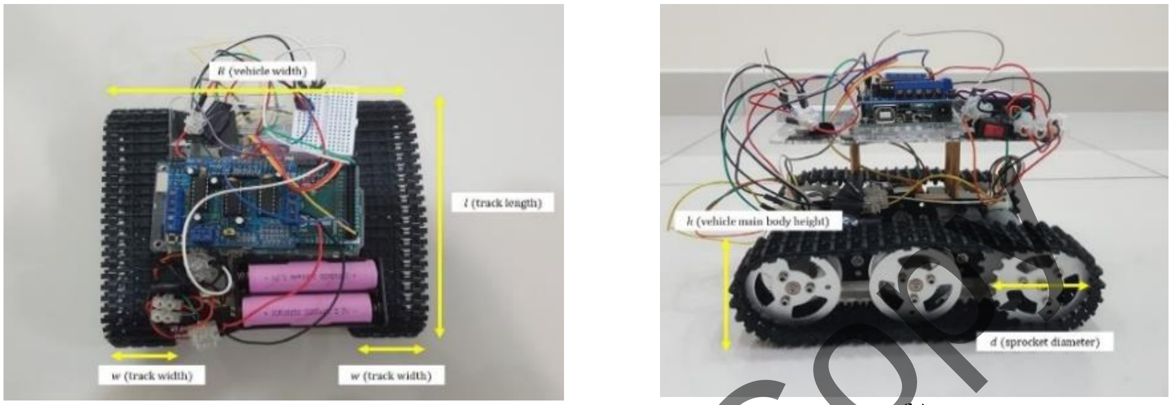
Fig. 4 Kinematics model of step crossing

Pre-Proof Copy

2.2 Developing a Virtual Testing Ground for Tracked Mobile Robots with Unreal Engine

This section explores the implementation of a skid-steering model in Unreal Engine. It includes the following approaches: 1) Defining the Car Character (Pawn Character), which sets the foundation for the robot's in-simulation existence, and 2) Blueprint Design, where we utilise Unreal Engine's visual scripting system to program the robot's movement and behaviour.

The TP100 Crawler Caterpillar was chosen as the experimental model. It is 185mm long, 200mm wide, and weighs 0.65 kg, with top and side views shown in Fig. 5. The system uses an Arduino Mega for its main electronics, connected to MPU6050 and encoder sensors for essential rotation and distance data. Motor functions are controlled by the L293D driver, with power from two 18650 lithium 3.7 V batteries.



(a) (b)

Fig. 5 Platform of TP100 Crawler: (a) View from the Top (b) View from the Side

Table 2 presents the specifications of the tracked car.

Table 1: Specifications of the tracked car

Item	Value and unit
Initial Mass	0.650 kg
Total Mass with Electrical Components	0.922 kg
Size (LxWxH)	200 mm x 185 mm x 60 mm
Wheel Size	24 mm
Track Width	45 mm
Track Length	185 mm

During the stage of defining the pawn character, the initial design phase involved creating a 3D model of the tracked mobile robot platform using SolidWorks®, a computer-aided design (CAD) software. Each component of the robot was modelled with precise real-world dimensions and subsequently assembled within a master assembly in SolidWorks® (refer to Fig. 6 for the visualisation).

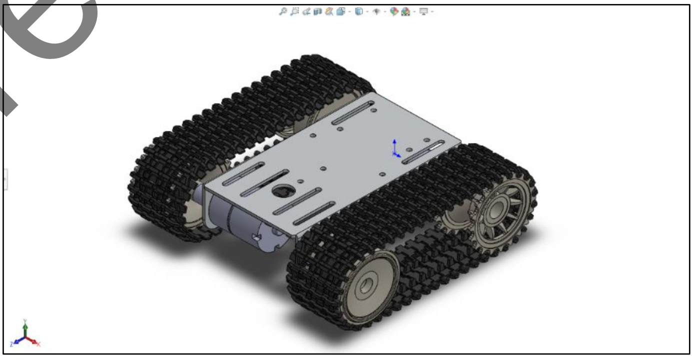


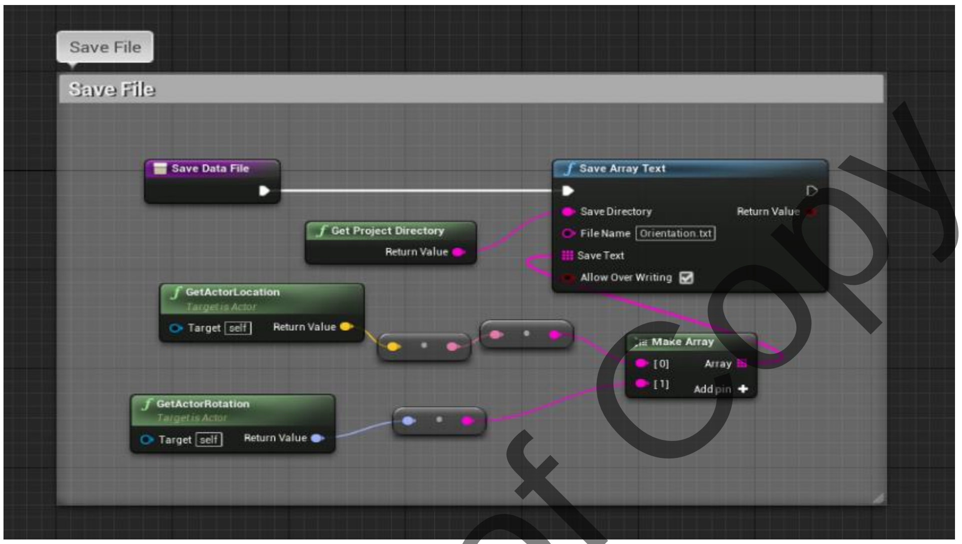
Fig. 6 The SolidWorks® CAD model of the TP100 Crawler Caterpillar

In Unreal Engine, functionality is mainly developed using the Blueprint Class, often called a Blueprint, during the Blueprint Design phase. Blueprints, either created via Blueprint Visual Scripting (a visual coding language) or C++ code, are objects or classes that operate by connecting nodes in a graph to perform actions and processes. These nodes can handle tasks such as creating objects, defining functions, and responding to events. Each Blueprint instance is customisable through its nodes, and when configured, it can be integrated into the virtual environment as an object instance. Any changes made to the original Blueprint automatically apply to all its instances within the environment [36].

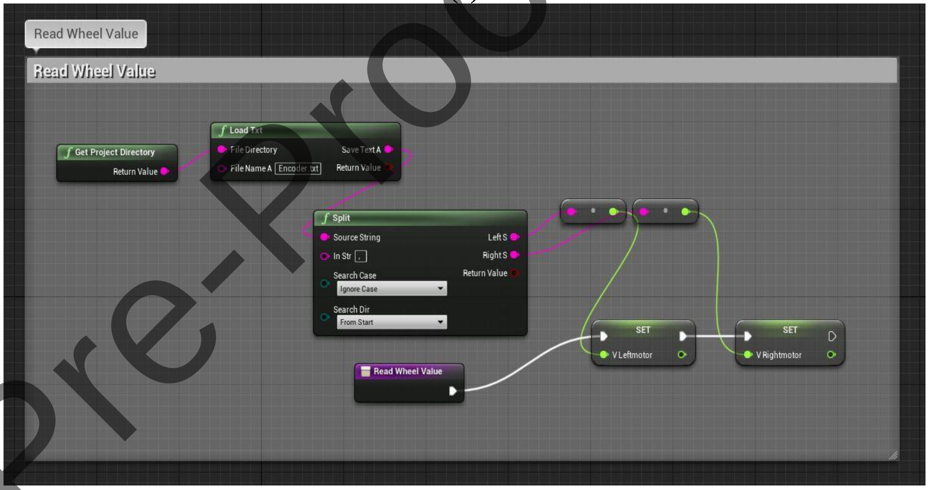
Pre-Proof Copy

For communication between Unreal Engine, the simulation space, and the Python-controlled robot, a method utilising text files for data exchange was used. This process involves inter-process communication (IPC), allowing for data sharing between programs. Two crucial text files facilitate this communication: 1) Encoder.txt, which records wheel speed data from the Arduino Mega shown in the Windows command prompt, and 2) Orientation.txt, which captures the simulated robot's position and orientation in Unreal Engine. The communication starts with reading the wheel speed from Encoder.txt and then feeding this into the Unreal Engine simulation. The simulation then processes this data using its kinematic model blueprint, detailed in Figure 7(a) and (b), which manages the reading and writing of these text files.

The kinematic model comprises two main functions, as previously derived in Figure 7(c) and (d): 1) Skid Steering Function, which manages the steering of the tracked mobile robot in the simulation, and 2) Obstacle Climbing Function, allowing the simulated robot to navigate walls or steps.

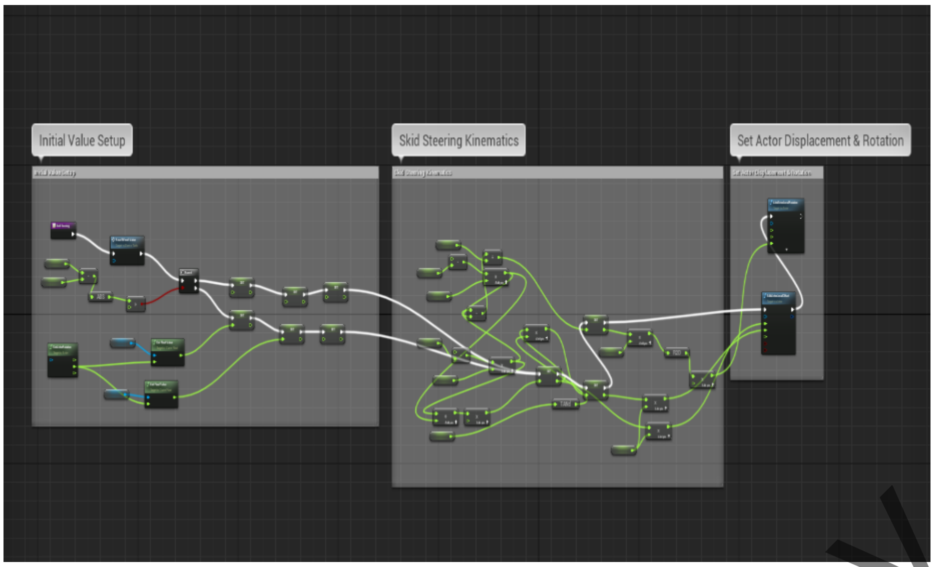


(a)

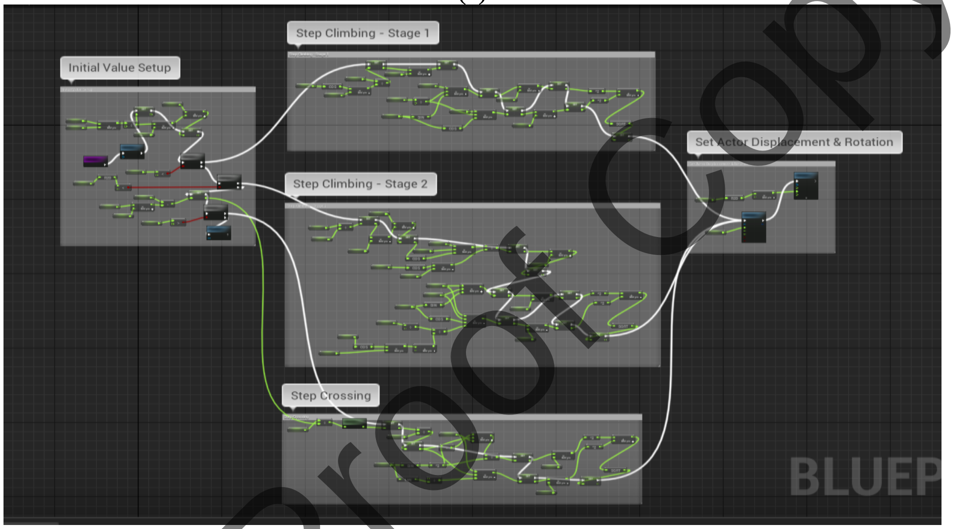


(b)

Pre-Proof Copy



(c)



(d)

Fig. 7 Blueprint Visual Scripting with (a) “Save Data File” (Writing Text) (b) “Read Wheel Value” (Read Text) (c) Skid Steering Model (d) Obstacle Climbing Model

Pre-Proof Copy

3.0 RESULTS AND DISCUSSION

3.1 Skid Steering Data Collection for Tracked Mobile Robot Model

As Dogru [37], [38] indicates, the equivalent wheelbase equation is derived from the schematic diagram of the tracked mobile robot presented in Fig. 8.

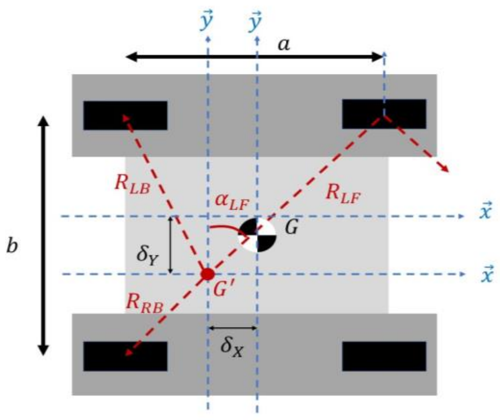


Fig. 8. Schematic of Skid-Steered Tracked Car

The method suggests that the robot rotates around its centre of mass, with closer wheels to this centre providing better traction. The equivalent wheelbase is calculated using the distances from the wheels to the centre of rotation, denoted as R_{LF}, R_{LB}, R_{RF} and R_{RB}, and the angles α_{LF}, α_{LB}, α_{RF} and α_{RB} between the y-axis and each distance vector. This leads to the wheelbase formula, which is stated as:

$$L = \frac{2\left(\left(\frac{b}{2} - \gamma\delta_y\right)^2 + \left(\frac{a}{2} - \delta_x\right)^2\right)}{\left(\frac{b}{2} - \gamma\delta_y\right)} \qquad (7)$$

Here, a and b are the robot’s length and width, while δx and δy are the offsets in the x and y directions from the robot’s centre of mass to its geometric centre. The factor γ adjusts δy, determined experimentally. These offsets are inferred from the robot’s weight distribution, with Table 3 presenting the wheelbase findings.

Table 2 Result of the equivalent wheelbase

Parameters	Values
Robot Geometry Length, a	13.3 cm
Robot Geometry Width, a	16 cm
X axis Offset Distance, δx	0.174 cm
Y axis Offset Distance, δy	-0.137 cm
Equivalent Wheel Base, L	27.42 cm

Slip parameters were evaluated using a tarpaulin surface to replicate slippage. Angular velocity and wheel speed were measured using an MPU6050 gyroscope and encoders, while an overhead camera system tracked the robot's trajectory. Tests revealed consistent slip ratios across straight and turning manoeuvers, with minor variations in slip angles. These parameters were integrated into the simulation model to refine its accuracy. An overhead camera system was used to trace the vehicle's speed and movement, improving the robot's indoor positioning [39]. The pose estimation utilised ArUco markers. The markers were selected because of its simplicity and fast processing, as each mark was represented by a unique binary pattern [40], [41]. For the experiment, the developed robot was equipped with ArUco markers. It is shaped as square fiducial markers that are easily detectable. In the experiment, these markers were placed into specific areas on the robot, which can be used to facilitate accurate pose estimation. As the patterns were unique, they allowed the system to differentiate different markers, and as a result, we used them to track multiple points on the robot simultaneously. This multi-point tracking is essential to determine precisely the position and orientation of the robot.

The developed robot was assessed in an 8ft by 4ft area, marked in 10cm squares, providing a reference for its position. A GoPro camera positioned 2.5 meters high in the area's centre captured the movement of the robot. The overhead camera system was deliberately placed to obtain the entire test area, which was marked in a grid pattern that can be applied to provide reference points for the position of the robot. Together with the ArUco markers, the overhead camera allowed the robot to be monitored in real-time. This enabled us to record both linear and angular displacements accurately.

The movement involved straight and turning actions, as illustrated in Fig. 9. The efficiency of the robot's motion was determined by analysing the slip ratio and angle from the recorded data. To ensure that the result is reliable, each manoeuvre type is repeated three times under the same conditions. Slip occurs when there is a difference between the commanded wheel velocity and the actual velocity because of surface friction and incline. By investigating the slip angle (the angle between the robot’s trajectory and the direction of its wheels) and the slip ratio (the difference between the commanded and actual wheel speeds), the study was able to improve the kinematic model, which was applied in the simulation. The system's ability to provide real-time feedback on the robot's performance played a pivotal role in validating the simulation model, ensuring that the virtual environment closely mirrored the robot's behaviour in the physical world.

Pre-Proof Copy

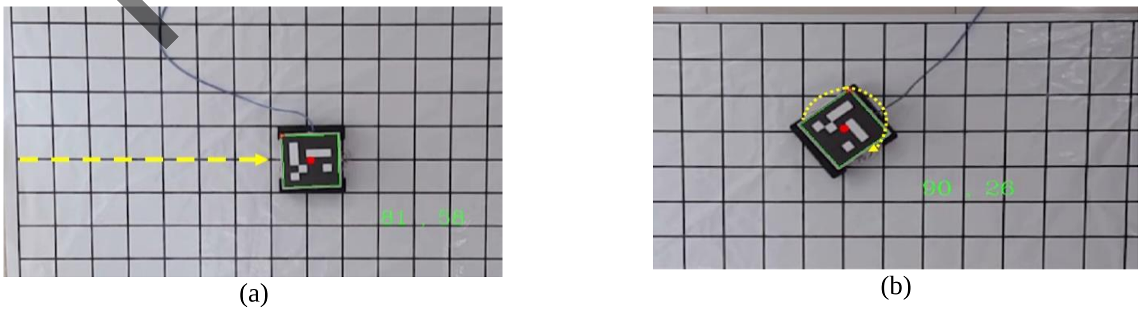


Fig. 9 Manoeuver ; (a) Straight (b) Turning

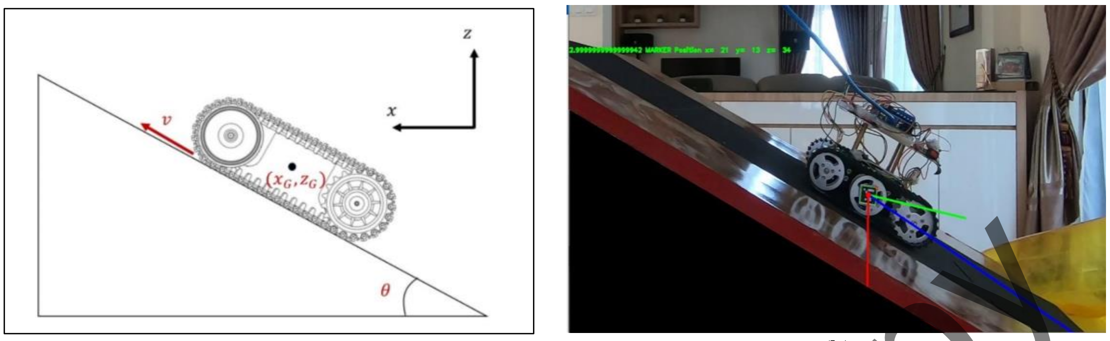
The results of the slip parameters are shown in Table 4.

Table 3 Obtained slip parameters from the experiment

Slip Parameters	Straight Manoeuver	Turning Manoeuver
Slip Angle, β	0.404°	0°

Left wheel Slip Ratio, α_l	0.1	0.1
Right wheel Slip Ratio, α_r	0.1188	0.1188

The experiments found that the slip ratios for both straight and turning manoeuvres were consistent for the left and right wheels, though the slip angles differed. These parameters (slip ratio and slip angle) are essential for the Hardware-in-the-Loop (HIL) simulations and will be applied in subsequent stages of the process. This section is dedicated to analysing skid-steering parameters for ramp climbing. Experiments on straight manoeuvres at various slope angles, depicted in Figure 10, were conducted. In these tests, the robot moved straight while inclined upwards (mimicking climbing), enabling the estimation of the slip ratio.



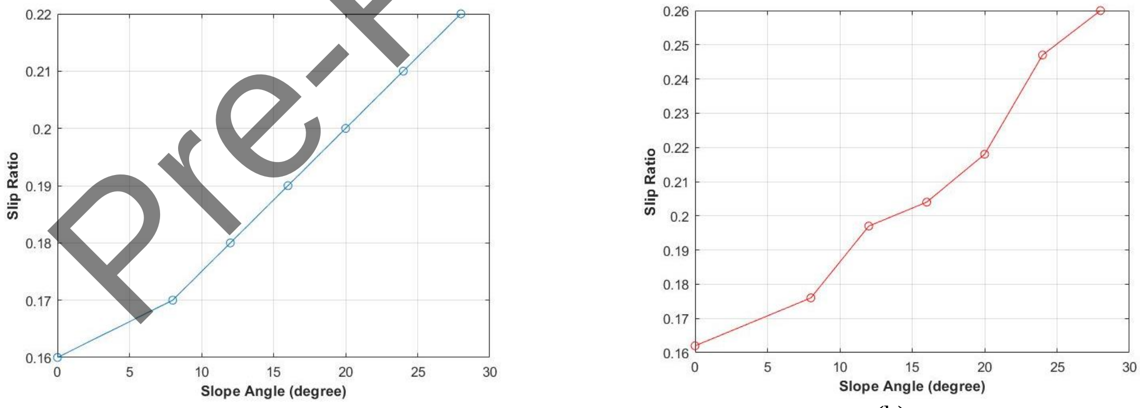
(a) (b)

Fig. 10 (a) Kinematics model of ramp climbing (b) Experiment of ramp climbing

In the experiment, the pitch angle and wheel velocity of the robot were measured with an MPU6050 gyroscope and encoders, respectively. A motion capture camera and an ArUco marker (ID = 72) were employed to determine the robot's position, with the marker placed at the centre of the left side wheel, as shown in Figure 12(b). The experiment involved setting the track velocity to 10 cm/s and assessing the tilt angle of the slope at increments of 4º, specifically at 8º, 12º, 16º, 20º, 24º, and 28º. Figure 11 displays the slip ratio results under climbing conditions at each slope angle, with the x-axis indicating the slope angle and the y-axis denoting the slip ratio.

Table 4 Result of experiment - slip ratio with respect to slope angle

Angle	**slip Left**	**slip Right**
0	0.16	0.162
8	0.17	0.176
12	0.18	0.197
16	0.19	0.204
20	0.2	0.218
24	0.21	0.247
28	0.22	0.260



(a) (b)

Fig. 11 Slip ratio vs slope angle ,(a) Left wheel (b) Right wheel

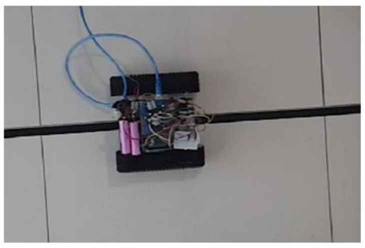
Fig. 11 demonstrates that the slip ratio during straight manoeuvring has a relatively linear relationship with the slope angle. The data collected from this experiment will be integrated into the Unreal Engine skid-steering model.

3.2 Performance Evaluation with Behaviour Test Course 1

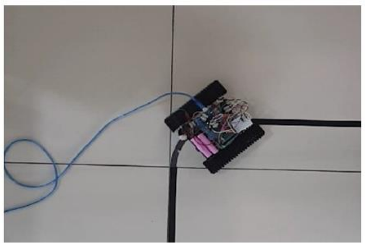
To assess the effectiveness of the proposed model, we employed a dedicated test course focusing on basic robot locomotion (Behaviour Test Course 1). This course aims to evaluate the robot's autonomous navigation capabilities and its ability to handle turns with varying radii. The course layout, as shown in Fig. 12, comprises three sections: (a) Straight Line: This section assesses the robot's capability to traverse a straight path. (b) Radius Curved Turns: This section tests

Pre-Proof Copy

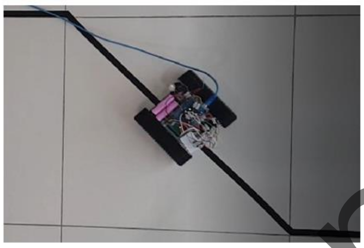
the robot's performance in making turns with specific, curved radii. (c) Sharp Turns: This section assesses the robot's handling accuracy during sharp turns [42].



(a)



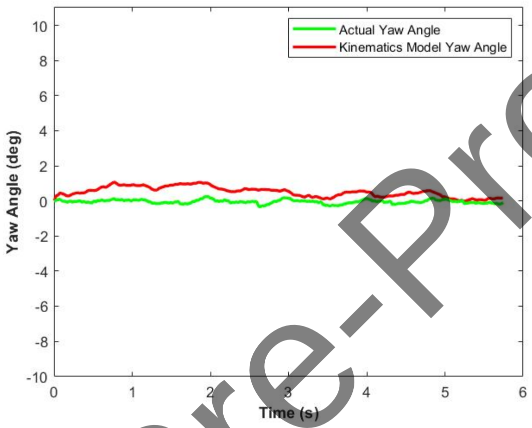
(b)



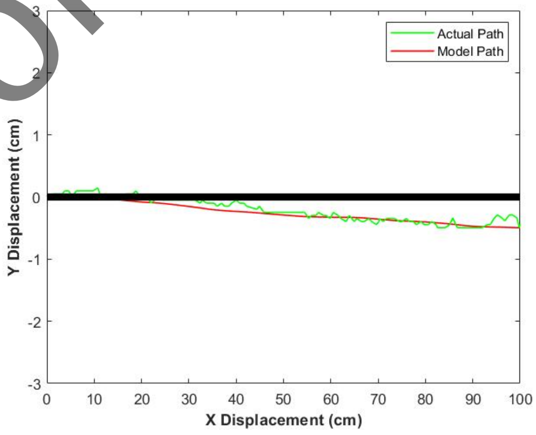
(c)

Fig. 12 Course Layout (a) Straight-line (b) Radius Curve (c) Sharp Turns

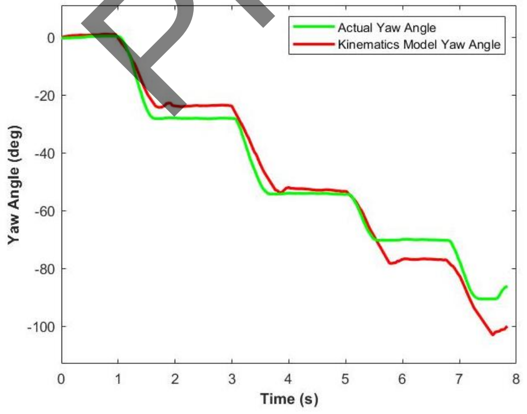
Fig. 13 presents the results of the simulation compared to the robot's actual performance. The green lines represent the ground truth, which refers to the robot's actual path (during locomotion) and actual yaw angle (during turns). The red lines represent the corresponding values obtained from the Unreal Engine kinematic model derivation, namely the model path and model yaw angle. This comparison allows us to evaluate how closely the simulation reflects the robot's real-world behaviour.



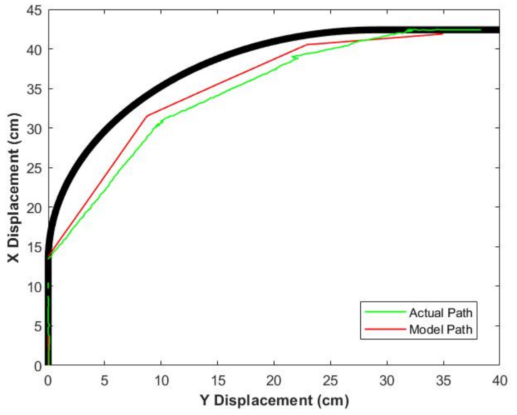
(a)



(b)

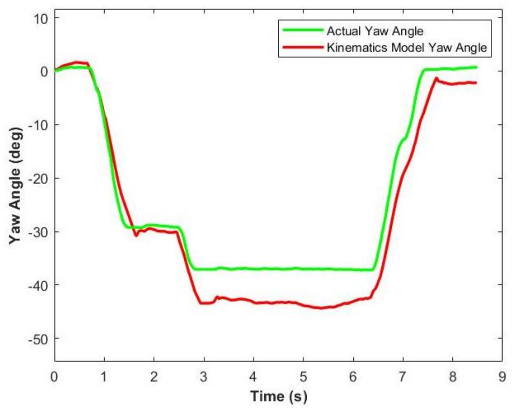


(c)

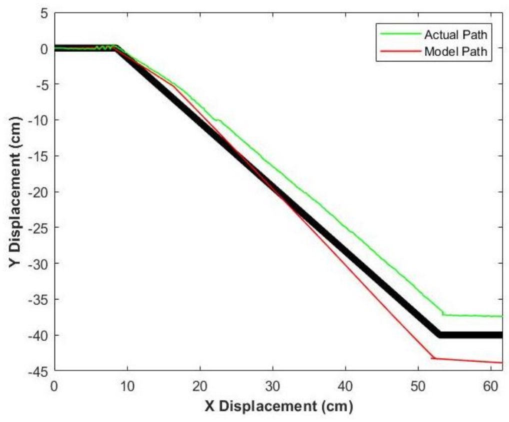


(d)

Pre-Proof Copy



(e)



(f)

Fig. 13 Straight-Line Course (a & b), Radius Curve Course (c & d) and Sharp Curve Course (e & f) Yaw Angle and Position Result

The results show that the direction angle is in close agreement with the ground truth. Nevertheless, variances exist between the estimated positions and the actual ground truth. The model demonstrates superior performance in radius curves and straight-line trajectories over sharp curve routes. The average position error for each trajectory is quantified using the following equation, where the positions of the actual path and model path are substituted:

$$avaerage\ position\ error = \sqrt{(x - x_c)^2 + (y - y_c)^2} \qquad [8]$$

The coordinates(x,y) and (x_c, y_c) represent the ground truth and estimated positions, respectively. A comparison of the maximum position errors across three test courses—straight line, radius curve, and sharp curve—was conducted. The straight-line course had an error of 1.5625 cm, the radius curve 1.2028 cm, and the sharp curve the largest at 8.5838 cm. Despite these errors, they are considered acceptable for our application. The errors observed in the straight-line, radius curve and sharp curve courses are considered acceptable for several reasons. The maximum position errors, such as 1.5625 cm in the straight-line course and 1.2028 cm in the radius curve course, are relatively minor and do not significantly impact the robot's ability to navigate and perform its tasks effectively. Given the nature of the test environment and the physical constraints of the tracked robot, these errors remain within a manageable range for real-world applications, such as search and rescue operations or military tasks, where slight deviations are tolerable and do not critically affect the overall mission.

3.3 Interactive Simulation Framework with Hardware-in-the-Loop Simulation

This section presents the hardware setup for a hardware-in-the-loop interactive simulation, illustrated in Figure 14's block diagram. The setup includes the actual tracked mobile robot, consisting of two primary elements: (a) the Arduino Mega Microcontroller, serving as the robot's central processing unit, executing the input commands, and (b) the Wheel Motors, tasked with the robot's locomotion.

In contrast, the simulation plant embodies the computer-generated virtual environment. It is constructed using two key software components: (a) the Python Command Prompt, utilised for issuing control commands to the robot and processing its sensor data, and (b) the Unreal Engine Software, a game engine used to replicate the robot's actions and its interactive virtual surroundings.

Communication between the Arduino Mega and the computer is established through a serial connection using a USB cable. The robot transmits its wheel speed data to the computer, and in return, the Unreal Engine simulation provides feedback on the robot's virtual position and orientation. This reciprocal communication facilitates the interaction between the physical robot and its virtual representation in the simulation.

Pre-Proof Copy

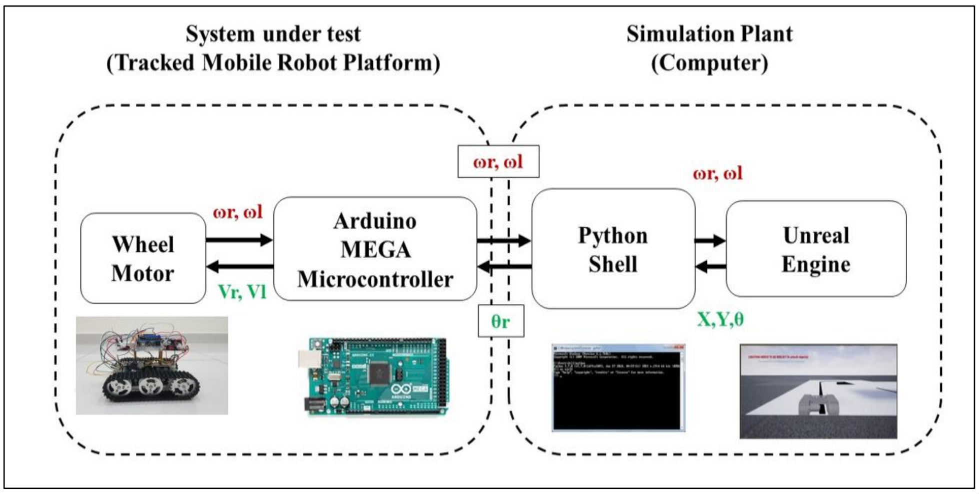


Fig 14 Interactive simulation with Hardware-In-The-Loop

The simulation serves a crucial purpose: testing and validating the effectiveness of a basic trajectory-tracking controller for the robot. This is achieved by creating a simulated environment that interacts with the real robot. A Python script plays a key role in initiating the HIL process. Fig.15 illustrates that the script executes the following step: 1) Data Acquisition: It collects the robot's position and orientation data from the Unreal Engine simulation. 2) Trajectory Control Integration: The script extracts the X and Y coordinates from the position data. These values are then fed into the trajectory tracking controller. 3) Desired Yaw Angle Calculation: Based on the robot's position within the simulated environment, the controller calculates the desired yaw angle (turning angle) required to follow the planned trajectory. This process essentially allows the controller to utilise the information from the virtual world to determine the necessary adjustments for the robot's movement in the real world.

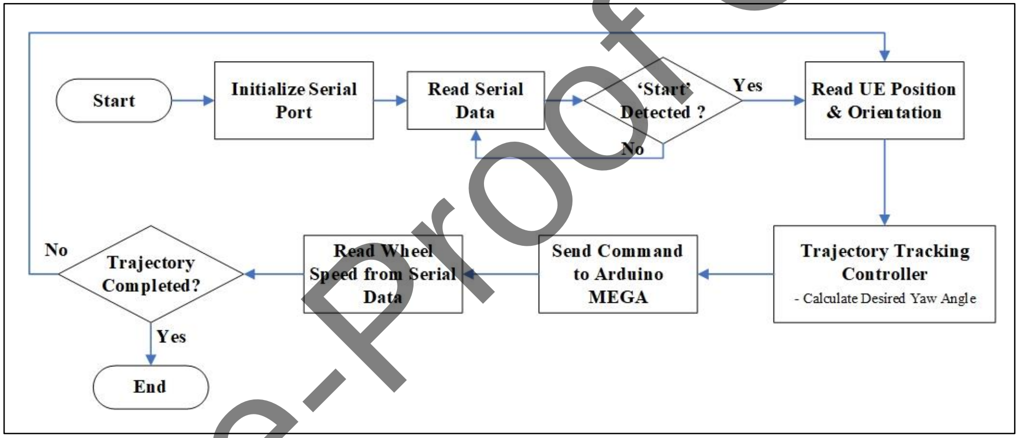

Start
Initialize Serial Port
Read Serial Data
'Start' Detected ?
Yes
No
Read UE Position & Orientation
Trajectory Tracking Controller
- Calculate Desired Yaw Angle
Send Command to Arduino MEGA
Read Wheel Speed from Serial Data
Trajectory Completed?
No
Yes
End

Fig. 15 The Flow Diagram

Two behaviour test courses were used in the interactive simulation to evaluate the robot's capabilities: 1) Behaviour Test Course 2, which tested the robot's reaction to minor disturbances and its error recovery ability during movement [42], and 2) Behaviour Test Course 3, aimed at examining the robot's performance in various locomotion modes, including skid steering, ramp climbing, and obstacle climbing. Both test courses were constructed within a virtual world mirroring the real-world testing environment, with an area of 1.9 meters x 1 meter (Fig. 16). This allowed us to validate the effectiveness of the robot's trajectory tracking control in a simulated setting. For Behaviour Test Course 3, a specific obstacle scenario was implemented within the simulation. A 3.7 cm tall obstacle was positioned at coordinates (120 cm, -80 cm), while a ramp with a 28-degree incline was placed at coordinates (70 cm, -80 cm). By introducing these elements, we could evaluate the robot's ability to navigate around obstacles and climb inclines.

Pre-Proof Copy

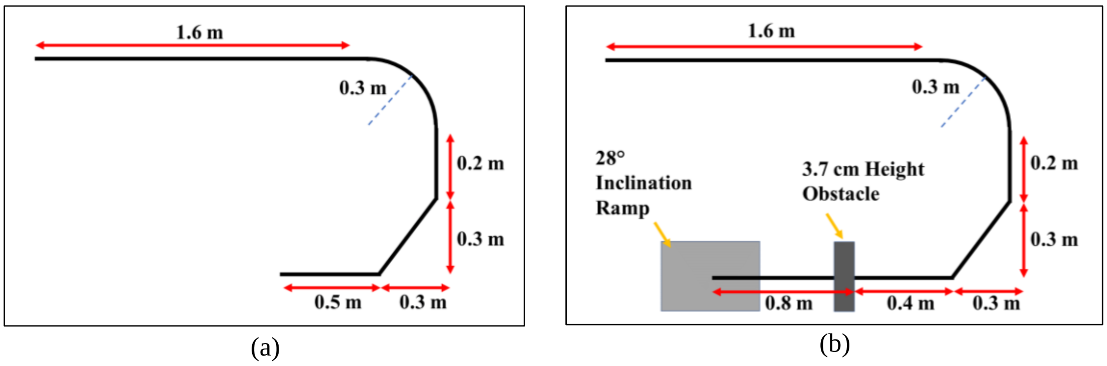


Fig. 16 (a) Behaviour Test Course 2 (b) Behaviour Test Course 3

Figures 17 and 18 display the robot's yaw angle and trajectory tracking during the HIL simulation, with green lines indicating actual behaviour and red lines showing Unreal Engine model predictions. In Behaviour Test Course 2, both real and virtual robots started at the trajectory's origin point (0,0). Initially, during straight-line and curved-radius movements, the robot accurately followed the real-world trajectory. After the sharp turn, it underestimated the travelled distance, with the yaw angle remaining close to the ground truth but with a significant position deviation, likely due to velocity estimation errors. In Behaviour Test Course 3, similar to Course 2, initial adherence to the trajectory was observed. However, during curved movements, slight rotational estimation errors led to divergence from the planned path. Despite these errors, the Unreal Engine model's yaw angle closely matched the ground truth, indicating effective rotational estimation from wheel encoder data.

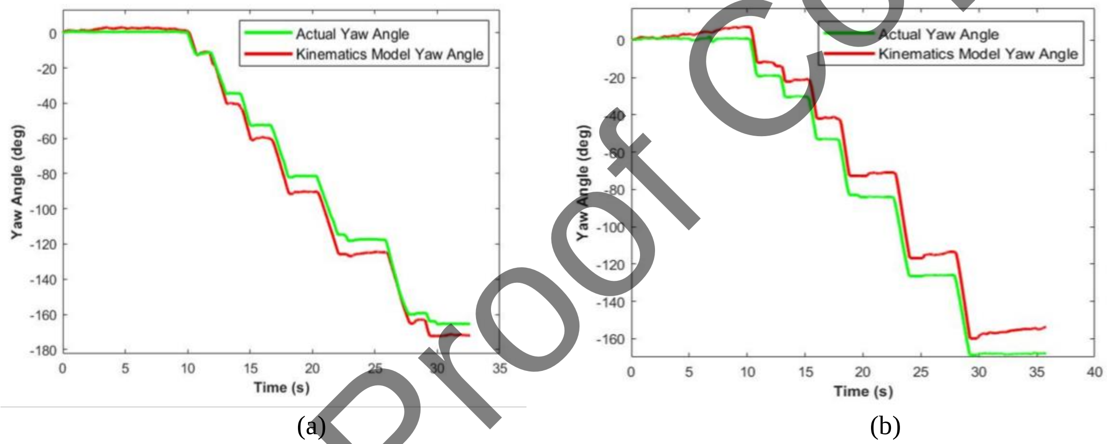


Fig. 17 Trajectory Tracking Control - Yaw Angle (a) Behaviour Test Course 2 and (b) Behaviour Test Course 3

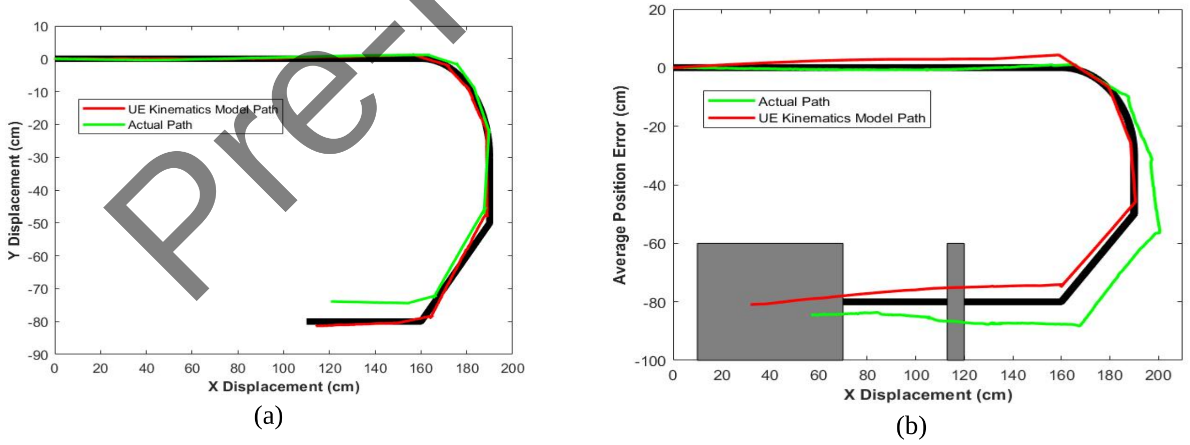


Fig. 18 Trajectory Tracking Control – Spatial displacement (a) Behaviour Test Course 2 (b) Behaviour Test Course 3

In the validation path, Table 7 summarises the mean squared performance metrics. The data show that the overall position error for both the x and y Cartesian coordinates is maintained within 1.7 meters, and the yaw angle error is limited to 0.83 radians.

Table 5 Mean Squared Error for Position and Yaw Angle (Behaviour Test Courses 2 and 3)

Mean Squared Error	**Δx (m)**	**Δy (m)**	**$\Delta\theta_{yaw}$ (rad)**
Behaviour Test Course 2	0.1854	0.1486	0.801

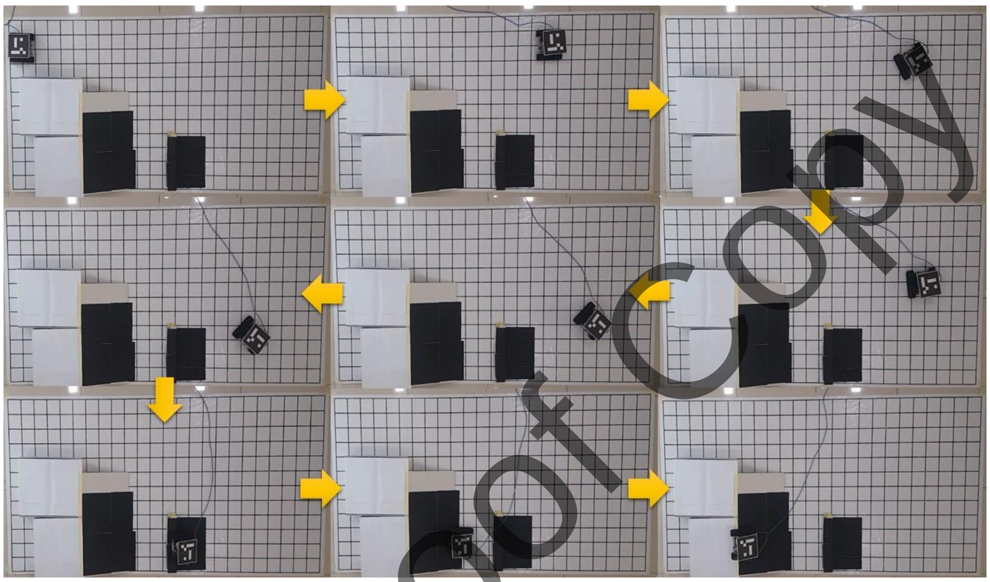
Pre-Proof Copy

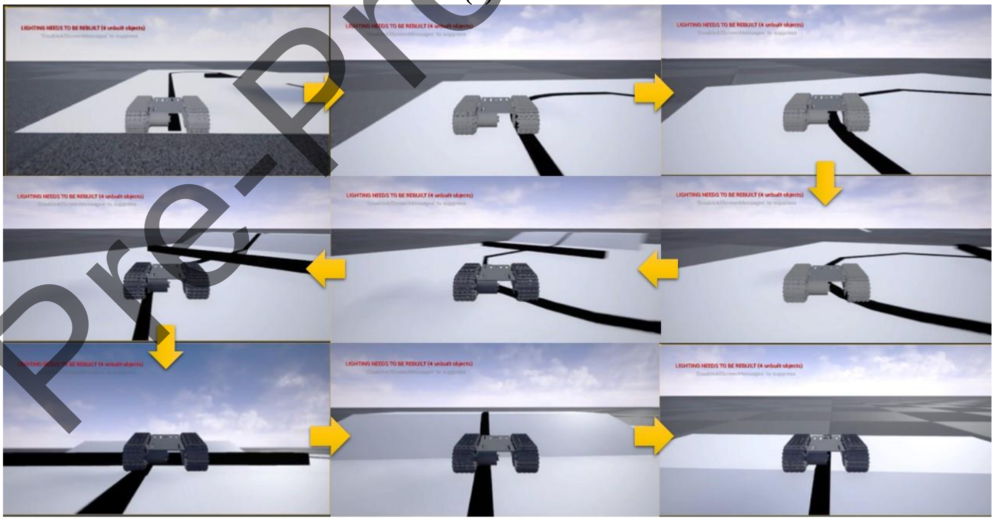
Behaviour Test Course 3	1.6607	0.4948	0.821

Comparative analysis of two test courses revealed superior robot performance in Test Course 2, with more accurate position estimation than in Test Course 3. This is supported by the mean squared error values listed in Table 7, which are markedly lower for Course 2. The increased positional error in Test Course 3 is likely due to an underestimation of the robot's longitudinal velocity, especially during obstacle navigation and ramp climbing, resulting in a maximum position deviation of 0.341 meters at the course's conclusion.

Despite the positional estimation challenges in Course 3, the overall outcomes were favourable. The implemented model and its trajectory tracking controller effectively piloted a U-shaped route across varied terrains, demonstrating the ability of the system to retain a predefined path with sensory feedback from the Unreal Engine simulation. The results show the importance and efficacy of interactive simulations for evaluating path-following controllers in an engineering context. Screenshots of video representation of the simulation process are provided below for further visualisation.

Pre-Proof Copy


(a)



(b)

Fig. 19 (a) Robot's Position (b) Unreal Engine Robot's Position (Behaviour Test Course 3)

Results show that the proposed system has accurate trajectory tracking for a tracked mobile robot. The yaw angle estimation aligns with the ground truth, indicating the effectiveness of the slip-compensated kinematic model and the developed sensor fusion techniques. Compared to existing studies, the mean squared errors for the position (0.1854 m, 0.1486 m) and yaw angle (0.801 rad, 0.821 rad) close with or outperform MATLAB Simulink-based simulations by leveraging real-time physics modelling in Unreal Engine. Studies utilising MATLAB Simulink for HIL simulations repeatedly report higher uncertainty in real-world deployments due to solver approximations and hardware timing issues [16], [18], [43]. In contrast, Unreal Engine uses real-time physics modelling. This will allow significant adaptability in dynamic environments.

The proposed framework has good potential as it presents a scalable and cost-effective testing environment, bridging the gap between simulation and real-world deployment. This is important, especially for specific applications such as search and rescue, military, and autonomous exploration. The framework will enable rapid prototyping and evaluation of navigation algorithms before deploying the developed robot in challenging situations. Despite these advantages, the simulation has several constraints. For example, while the slip-compensated kinematic model improves the trajectory estimation, improvements in terrain-adaptive control strategies are required to handle complex situations. Since the framework was developed with a relatively small robot (185 mm length, 0.922 kg mass), further validation with larger autonomous tracked systems is required to confirm the scalability of the proposed framework.

To conclude, the study shows the potential of Unreal Engine for real-time simulation, which enables a high level of pragmatism in robot-environment interactions in comparison to traditional kinematic simulators. Furthermore, the integration of hardware and software under the HIL framework ensures that the system can adapt to real-world constraints. This shows that the proposed approach can be a practical tool for engineers in designing robots for complex environments

4.0 CONCLUSION

In the study, we develop an innovative Hardware-in-the-Loop (HIL) simulation framework that is fitted for tracked mobile robots. By improving and incorporating customised and detailed kinematic models with Unreal Engine, the framework is able to address the gap between virtual simulations and real-world adoption. In the project, the kinematic models for skid-steering and obstacle climbing were implemented and validated through comprehensive testing. The developed approach was adopted for a tracked robot and tested in several test courses. Two behaviour test courses were run to evaluate the model: 1) Behaviour Test Course 2, which tested the robot's reaction to minor disturbances and its error recovery ability during movement and 2) Behaviour Test Course 3, aimed at examining the robot's performance in various locomotion modes, including skid steering, ramp climbing, and obstacle climbing. Experiment results showed a maximum position error of ±8.5838 cm and mean squared errors of 0.1854 m (x-coordinate), 0.1486 m (y-coordinate), and 0.801 radians (yaw angle) in Behaviour Test Course 2. Behaviour Test Course 3 demonstrated slightly higher errors with mean squared values of 1.6607 m (x-coordinate), 0.4948 m (y-coordinate), and 0.821 radians (yaw angle). These findings emphasise the simulation framework's ability to closely replicate real-world robot performance, particularly in flat and curved terrain scenarios.

The framework successfully reflected the physical behaviour of the tracked robot throughout various circumstances, exhibiting the utility of its kinematic models and Hardware-in-the-Loop (HIL) integration. The approach improves the difference between virtual simulations and real-world testing, offering a valuable tool for engineers and researchers. The ability to calculate and predict the behaviour of the robot in customised environments has significant implications for applications such as urban search and rescue, military operations, and autonomous vehicles, where reliable navigation is vital.

Despite its effectiveness, the study discovered some constraints of the developed approach, such as dependence on a specific robot model and controlled test conditions. Future work will address these by incorporating various robot designs, extensive terrain types, and additional environmental factors to improve the framework's robustness and versatility.

The developed framework supports the full customisation of kinematic models and real-time HIL integration. This approach will enable precise simulation of unconventional locomotion systems, such as skid-steering tracked robots. Unlike traditional platforms, the framework provides real-time feedback between physical hardware and virtual environments, minimising discrepancies between simulated and real-world performance.

This research can provide a substantial contribution to mobile robotics, offering a flexible and precise platform for real-time robot simulation and testing. It has a good potential for broader applications and can be used to address the need for further investigation and development to fully realise its capabilities in real-world scenarios. In future work, we aim to reduce the current maximum position error of ±8.5838 cm by improving sensor calibration to minimise drift and noise in positional data. Advanced localisation techniques, such as sensor fusion from IMUs and overhead camera systems, will be implemented to enhance real-time pose estimation. Refinements to the kinematic model, including improved slip compensation and dynamic modelling, will better account for wheel slippage and terrain friction. Optimising trajectory tracking algorithms is another key focus to ensure minimal path deviations.

Additionally, we plan to compare our framework with other simulators to benchmark its effectiveness. Expanding tests to include diverse terrains, such as loose gravel, sand, and slopes, as well as a variety of obstacle types, will validate the framework’s adaptability and robustness under real-world conditions.

In conclusion, expanding the simulation fidelity in Unreal Engine by combining more detailed environmental physics will further reduce the discrepancies between simulated and real-world performance. These enhancements are supposed to considerably amend the framework’s accuracy and utility for robotics applications.

5.0 CONFLICT OF INTEREST

The authors declare no conflicts of interest.

Pre-Proof Copy

6.0 AUTHORS CONTRIBUTION

All authors contribute to the manuscript. Loh Chow Yeh (Investigation; Software; Resources). Vimal Rau Aparow (Validation; Writing -editing). Hermawan Nugroho (Writing -original draft and editing).

7.0 ACKNOWLEDGEMENTS

The authors would like to thank the University of Nottingham Malaysia for financial support of this project.

Funding

This study was not supported by any grants from funding bodies in the public, private, or not-for-profit sectors.

8.0 REFERENCES

[1] L. Bruzzone, S. E. Nodehi, and P. Fanghella, "Tracked Locomotion Systems for Ground Mobile Robots: A Review," *Machines*, vol *10*(8), 6482022, pp. 648, 2022

[2] R. Bsc, U. Pal, S. P. Ojha, V. Srinivasan, and A. Chakrabarti, "Exploring serially connected multi-tracked all-terrain vehicles for improved obstacle climbing performance," in *14th National Conference on Machines and Mechanisms, NaCoMM 2009*, 2020, pp. 279-286.

[3] G. Dimitrakopoulos, A. Tsakanikas, and E. Panagiotopoulos. Autonomous Vehicles Technologies, Regulations, and Societal Impacts, 1st ed. Elsevier Science, 2021.

[4] Z. Szalay, "Next generation X-in-the-loop validation methodology for automated vehicle systems," *IEEE Access*, vol. 9, pp. 35616-35632, 2021.

[5] Y. Chen, S. Chen, T. Zhang, S. Zhang, and N. Zheng, "Autonomous Vehicle Testing and Validation Platform: Integrated Simulation System with Hardware in the Loop∗," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, China, 2018, pp. 949-956.

[6] X. Hu, "Applying robot-in-the-loop-simulation to mobile robot systems," in *2005 International Conference on Advanced Robotics, ICAR '05, Proceedings*, Seattle, WA, USA, 2005, pp. 506-513.

[7] R. P. Urvina, C. L. Guevara, J. P. Vásconez, and A. J. Prado, "An Integrated Route and Path Planning Strategy for Skid–Steer Mobile Robots in Assisted Harvesting Tasks with Terrain Traversability Constraints," *Agriculture (Switzerland)*, vol. 14, no. 8, pp. 1206, 2024.

[8] L. Liang *et al.*, "Model-Based Coordinated Trajectory Tracking Control of Skid-Steer Mobile Robot with Timing-Belt Servo System," *Electronics*, vol. 12, no. 3, pp. 122, 2023.

[9] J. Moreno, E. Slawiñski, F. A. Chicaiza, F. G. Rossomando, V. Mut, and M. A. Morán, "Design and Analysis of an Input–Output Linearization-Based Trajectory Tracking Controller for Skid-Steering Mobile Robots," *Machines*, vol. 11, no. 11, pp. 988, 2023.

[10] M. Akçakoca *et al.*, "A simulation-based development and verification architecture for micro uav teams and swarms," in *AIAA Scitech 2019 Forum*, San Diego, 2019, pp. 1917-2305.

[11] M. Nithya and M. R. Rashmi, "Gazebo - ROS - Simulink Framework for Hover Control and Trajectory Tracking of Crazyflie 2.0," in *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, Kochi, India, 2019, pp. 649-653.

[12] Z. B. Rivera, M. C. De Simone, and D. Guida, "Unmanned ground vehicle modelling in Gazebo/ROS-based environments," *Machines*, vol. 7, no. 2, pp. 42, 2019.

[13] H. Huang, H. Xu, F. Chen, C. Zhang, and A. Mohammadzadeh, "An Applied Type-3 Fuzzy Logic System: Practical Matlab Simulink and M-Files for Robotic, Control, and Modeling Applications," *Symmetry*, vol. 15, no. 2, pp.475, 2023.

[14] A. O. Prasad *et al.*, "Design and development of software stack of an autonomous vehicle using robot operating system," *Rob Auton Syst*, vol. 161, pp. 104340, 2023.

[15] D. T. Tran, T. D. Nguyen, M. K. Tran, and K. K. Ahn, "Development of a Hardware-in-the-Loop Platform for a Teleoperation Flexibility Robotic System," *Applied Sciences*, vol. 14, no. 5, pp. 2207, 2024.

[16] U. Davalos-Guzman, C. E. Castañeda, L. M. Aguilar-Lobo, and G. Ochoa-Ruiz, "Design and implementation of a real time control system for a 2dof robot based on recurrent high order neural network using a hardware in the loop architecture," *Applied Sciences* , vol. 11, no. 3, pp. 1–16, 2021.

[17] M. Tristano *et al.*, "Hardware-in-the-Loop Real-Time Implementation of a Vehicle Stability Control Through Individual Wheel Torques," *IEEE Trans Veh Technol*, vol. 73, no. 4, pp. 4683–4693, 2024.

[18] C. C. Dosoftei, A. T. Popovici, P. R. Sacaleanu, P. M. Gherghel, and C. Budaciu, "Hardware in the loop topology for an omnidirectional mobile robot using Matlab in a robot operating system environment," *Symmetry*, vol. 13, no. 6, pp. 969, 2021.

[19] B. Sebastian and P. Ben-Tzvi, "Physics Based Path Planning for Autonomous Tracked Vehicle in Challenging Terrain," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 95, no. 2, pp. 511–526, 2019.

[20] M. Abbas and S. K. Dwivedy, "Adaptive control for networked uncertain cooperative dual-arm manipulators: an event-triggered approach," *Robotica*, vol. 40, no. 6, pp. 1951-1978, 2022.

[21] C. Liu, Y. Yang, C. Zhang, and S. Yang, "Project-Based Teaching in Control Theory Education Based on V-REP: A Cart Inverted Pendulum Case," *Journal of Contemporary Educational Research*, vol. 7, no. 4, pp. 18-24, 2023.

Pre-Proof Copy

[22] K. Shabalina, A. Sagitov, K.-L. Su, K.-H. Hsia, and E. Magid, "Avrora Unior Car-like Robot in Gazebo Environment," in *Proceedings of International Conference on Artificial Life and Robotics*, Oita, Japan, 2019, pp. 116-120.

[23] J. Platt and K. Ricks, "Comparative Analysis of ROS-Unity3D and ROS-Gazebo for Mobile Ground Robot Simulation," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 106, no. 4, pp. 80, 2022.

[24] J. L. Silva Cotta, J. Rakoczy, and H. Gutierrez, "Precision landing comparison between smartphone video guidance sensor and IRLock by hardware-in-the-loop emulation," *CEAS Space Journal*, vol. 16, no. 4, pp. 475–489 , 2024.

[25] S. Sahu and A. K. Deb, "Swarm of Quadcopters Position control in ROS Gazebo Simulator," in *3rd International Conference on Range Technology, ICORT 2023*, Chandipur, Balasore, India, 2023, pp. 1-6.

[26] O. U. Acar, L. Güvenç, and E. Altuğ, "Hardware-in-the-Loop Testing of Automatic Lift Dropping System for Heavy Trucks," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 98, no. 3–4, pp. 693–703, 2020.

[27] R. Raveendran, K. B. Devika, and S. C. Subramanian, "Brake fault identification and fault-tolerant directional stability control of heavy road vehicles," *IEEE Access*, vol. 8, pp. 169229-169246, 2020.

[28] V. Kumar, S. C. Subramanian, and R. Rajamani, "Autonomous Emergency Braking of a Heavy Road Vehicle Using a Low-Density Flash Lidar," *IEEE Trans Veh Technol*, vol. 73, no. 2, pp. 1879-1889, 2024.

[29] R. Khan, F. M. Malik, A. Raza, and N. Mazhar, "Comprehensive study of skid-steer wheeled mobile robots: development and challenges," *Industrial Robot*, vol. 48, pp. 142-156 , 2021.

[30] G. Rigatos, "A Nonlinear Optimal Control Approach for Tracked Mobile Robots," *J Syst Sci Complex*, vol. 34, pp. 1279–1300, 2021.

[31] H. Li, H. Liu, J. Gai, and X. Li, "Steering Control of Dual-motor Coupling Drive Tracked Vehicle Based on PSO PID Parameter Optimization," *Binggong Xuebao/Acta Armamentarii*, vol. 45, no. 3, pp. 916-924, 2024.

[32] J. Gai, C. Liu, C. Ma, and H. Shen, "Steering Control of Electric Drive Tracked Vehicle Considering Tracks' Skid and Slip," *Binggong Xuebao/Acta Armamentarii*, vol. 42, no. 10, pp. 2092-2101, 2021.

[33] G. Yamauchi, K. Nagatani, T. Hashimoto, and K. Fujino, "Slip-compensated odometry for tracked vehicle on loose and weak slope," *ROBOMECH Journal*, vol. 4, no. 1, pp. 27, 2017.

[34] J. Liu, Y. Wang, S. Ma, and B. Li, "Analysis of stairs-climbing ability for a tracked reconfigurable modular robot," in *Proceedings of the 2005 IEEE International Workshop on Safety, Security and Rescue Robotics*, Kobe, Japan, 2005, pp. 36-41.

[35] A. H. Rajabi, A. H. Soltanzadeh, A. Alizadeh, and G. Eftekhari, "Prediction of obstacle climbing capability for tracked vehicles," in *9th IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2011*, Kyoto, Japan, 2011, pp. 128-133.

[36] A. R. G. Harwood, P. Wenisch, and A. J. Revell, "A real-time modelling and simulation platform for virtual engineering design and analysis," in *Proceedings of the 6th European Conference on Computational Mechanics: Solids, Structures and Coupled Problems, ECCM 2018 and 7th European Conference on Computational Fluid Dynamics*, , Glasgow, UK, 2018, pp. 1-8.

[37] S. Dogru and L. Marques, "An improved kinematic model for skid-steered wheeled platforms," *Auton Robots*, vol. 45, no. 2, pp. 229–243, 2021.

[38] M. F. Jaramillo-Morales, S. Dogru, and L. Marques, "Generation of Energy Optimal Speed Profiles for a Differential Drive Mobile Robot with Payload on Straight Trajectories," in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2020*, Abu Dhabi, United Arab Emirates, 2020, pp. 136-141.

[39] S. Adinandra and A. Syarif, "A low cost indoor localisation system for mobile robot experimental setup," *Journal of Physics: Conference Series*, vol. 1007, 2018.

[40] M. Irfan, S. Dalai, K. Kishore, S. Singh, and S. A. Akbar, "Vision-based Guidance and Navigation for Autonomous MAV in Indoor Environment," in *2020 11th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2020*, Kharagpur, India, 2020, pp. 1-5.

[41] M. F. Sani and G. Karimian, "Automatic navigation and landing of an indoor AR. Drone quadrotor using ArUco marker and inertial sensors," in *1st International Conference on Computer and Drone Applications: Ethical Integration of Computer and Drone Technology for Humanity Sustainability, IConDA 2017*, Kuching, Malaysia, 2017, pp. 102-107.

[42] W. R. Norris and A. E. Patterson, "System-level testing and evaluation plan for Field Robots: A Tutorial with Test Course Layouts," *Robotics*, vol. 8, no. 4, pp. 83, 2019.

[43] F. Mihalič, M. Truntič, and A. Hren, "Hardware-in-the-Loop Simulations: A Historical Overview of Engineering Challenges," *Electronics*, vol 11(15), pp. 2462, 2022.

Pre-Proof Copy