Universal Access in the Information Society (2026) 25:44
https://doi.org/10.1007/s10209-026-01308-9

LONG PAPER

Realistic simulation for dataset generation in a mobile robotics educational context

Laiany Brancaliao[1,2,3] · Mariano Alvarez[1,4] · J. A. B. Coelho[1] · Miguel Conde[2] · Paulo Costa[3,4] · Jose Goncalves[1]

Received: 11 July 2025 / Accepted: 8 January 2026

Abstract

In the context of mobile robotics education, realistic and accessible datasets are fundamental for supporting the development and testing of algorithms. However, collecting real-world data is a limited and challenging task because it is time-consuming and error-prone. Therefore, this paper presents the generation of a synthetic dataset through realistic simulation using the SimTwo environment—a physics-based simulator, and modeling techniques of sensors and actuators. The physical and simulated mobile robot was developed to perform tasks such as following a line, following a wall, and avoiding obstacles. The proposed approach facilitates the creation of customized datasets for training and evaluation algorithms while supporting remote and inclusive learning. Results show that a simulated dataset can effectively replicate real-world behaviors, making them a valuable resource for educational contexts, research, and development. Some emergent machine learning algorithms can be applied to this dataset, being this approach increasingly used to enhance robot localization, by leveraging ML, robots can improve the accuracy, robustness, and adaptability of their localization systems, especially in complex and dynamic environments.

Keywords Mobile robotics · Realistic simulation · Dataset generation · STEM education

Mariano Alvarez, J. A. B. Coelho, Miguel Conde, Paulo Costa and Jose Goncalves have contributed equally to this work.

✉ Laiany Brancaliao
laiany@ipb.pt

Mariano Alvarez
marianoalvarez@ipb.pt

J. A. B. Coelho
joaocoelho@ipb.pt

Miguel Conde
mcong@unileon.es

Paulo Costa
paco@fe.up.pt

Jose Goncalves
goncalves@ipb.pt

[1] Research Centre in Digitalization and Intelligent Robotics (CeDRI), Associated Laboratory for Sustainability and Technology in Mountain Regions (SusTEC), Polytechnic Institute of Braganca, Campus Santa Apolonia, 5300-253 Braganca, Portugal

[2] University of Leon, Campus Vegazana, 24071 Leon, Spain

[3] Institute of Engineering Systems and Computers, Technology and Science (INESC TEC), University of Porto, 4200-465 Porto, Portugal

[4] University of Porto, 4200-465 Porto, Portugal

1 Introduction

The application of autonomous mobile robot systems is constantly expanding due to the ability to assist in numerous tasks, from those applied in industry to daily life. The range of application areas for autonomous mobile robots is vast, including education, industry, personal assistance, daily tasks, agriculture, transportation logistics, construction, disaster management, exploration, surveillance, and sports entertainment [1, 2].

Several methods are applied to guarantee the mobile robots' effectiveness, including robot perception, control, decision-making, planning, localization, and others. The application of different kinds of sensors and also the combination of them through sensor fusion methods is widely applied in recent research [1].

The technological advances based on artificial intelligence have also contributed to improve the mobile robots' performance, decision-making, and self-learning [3]. Techniques such as supervised, unsupervised and reinforcement learning have been applied in environment mapping, simultaneous localization and mapping (SLAM), trajectory planning and others. Based on robot perception, acquired

Published online: 12 February 2026

by the sensors data, the mobile robots are able to adapt their behavior and decision-making, and consequently improving their navigation capacity and interaction with the environment [4, 5].

Regarding the control of mobile robots inside dynamic and unknown environments, another approach that has stood out is the deep reinforcement learning. This approach significantly minimizes the need for explicit modeling and manual programming, allowing robots to learn navigation strategies through direct interaction with the environment. A recent study highlighted the effectiveness of an enhanced proximal policy optimization (PPO) algorithm to guide mobile robots autonomously, using LiDAR sensor data to avoid obstacles and reach target destinations safely [6, 7].

Companies such as Boston Dynamics develop humanoid and mobile robots to perform complex tasks autonomously, applying computer vision and AI algorithms in order to understand the environmental conditions and adapt itself to it. Recent advancements in machine learning have enhanced the autonomy and versatility of mobile robots. Rather than relying solely on traditional algorithms, modern robotic systems now incorporate deep learning models that process both visual and linguistic data [8, 9].

Therefore, the development of autonomous systems, such as mobile robots, requires extensive testing and substantial data to ensure effective performance and continuous improvement. In this context, advanced AI-based algorithms have shown great potential for improving the performance of robotic systems. However, they depend on large volumes of high-quality data collected from diverse real-world scenarios, which is a complex and challenging task [10, 11].

Due to the complexity of collecting real-world data, mainly in dynamic and uncertain environmental conditions, in situations where the data collection is limited, difficult, unfeasible, time-consuming, and error-prone, simulated and synthetic data are emerging as an alternative solution for robotics systems development and improvements. Through simulated data, it is possible to create large and diverse datasets encompassing different environmental conditions and scenarios, which is an essential factor for training machine learning models [10, 12–14].

Recent research demonstrated the application of synthetic data, generated through artificial scenarios created in realistic simulations, as a reliable alternative to obtain real-world data. One of the advantages offered by synthetic datasets is the scalability, automatically creating an accurate ground truth without the possibility of human error, besides providing less dataset bias [10, 12, 15–18]. However, open datasets for public use are still scarce in some areas, as discussed by [19].

Nowadays, most of the training and testing processes for robotics navigation are learned in simulated environments, which must be close to reality [15]. According to [17], there is software applied to generate synthetic images to contribute to the training process of pose estimation of robotics tasks. In [20], the robot operating system (ROS) and Gazebo Simulator are applied to generate synthetic image datasets for machine learning algorithms. In paper [16], the virtual environment of the Grand Theft Auto V (GTA V), which is a realistic video game, was applied to create a dataset for robotics and navigation.

Synthetic datasets play a key role in accelerating machine learning development while improving the generalization of robotic models. These algorithms are exposed to situations that are often expensive and/or difficult to recreate in the physical world. The THUD++ dataset combines synthetic and real-world data to simulate dynamic indoor environments, offering more than 90,000 annotated image frames for tasks such as 3D object detection and autonomous navigation. Currently, platforms such as UnrealROX allow users to control virtual robots in immersive simulations, enabling the capture of sensor data and interactive behaviors. These tools are becoming essential for building mobile robots that can reliably operate in uncertain and unstable real-world conditions [21, 22].

In the context of educational robotics, these datasets have become increasingly valuable because provide accessible, safe, and cost-effective resources for the teaching process. Through the use of these synthetic datasets, students and educators are allowed to explore complex robotic behaviors and machine learning techniques without the need for expensive hardware or risk of damaging physical robots. In addition, synthetic datasets enable the simulation of diverse scenarios and environments that usually are difficult to reproduce in a typical classroom. This flexibility allows students to experiment complex tasks such as perception, path planning, and decision-making in a risk-free and scalable way. As a result, synthetic datasets support hands-on learning and innovation, helping to prepare the next generation of students with practical skills and theoretical knowledge [23, 24].

Generating these datasets through a simulation close to reality, the modeling of the sensors and actuators of the robot is indispensable as it provides an accurate representation of the behavior and dynamics of these subsystems. The modeling process is also relevant for sensor fusion, increasing the realism of simulations and supporting the development of high-performance controllers for mobile robots operating in dynamic environments [25–27].

As discussed, the design and development of modern systems require extensive testing to achieve reliability and high performance. Besides the appearance of synthetic datasets, the Hardware-in-the-loop (HIL) technique also stands out in industrial and educational contexts. In the HIL

approach, the physical hardware is integrated into the simulation environment, allowing testing control algorithms in real hardware conditions safely and efficiently. Compared to traditional development and testing methods, HIL can significantly reduce development time and costs, while enabling the early detection and correction of faults during the initial stages of system design. In educational contexts, HIL allows students to test their control algorithms on real hardware, even in situations where physical equipment is limited [28–30].

Therefore, the goal of this paper is to present the development of a realistic simulation inside an educational context in order to generate a dataset focused on mobile robots to publicly release so that researchers, companies, and research centers can leverage them, for example, to train machine learning algorithms, contributing to the advancement of robotic systems. The paper also describes the development of the physical mobile robot, prototyped with differential kinematics, which was applied in an educational context by master's students. This same robot is replicated in a virtual robot inside a realistic simulation to generate datasets focused on robot sensors and ground truth data, that can be used for localization, by applying, for example, machine learning algorithms, as well as classical algorithms, based on Kalman and particle filters.

In the following Sect. 2, the kinematics of a differential mobile robot are described, and then the mobile robot prototype is presented in Sect. 3. The control system design is described in Sect. 4, including the line following, wall following, and obstacle avoidance algorithms. The modeling process of sensors and actuators is presented in Sect. 5, followed by the simulation environment in Sect. 6. Section 7 presents the software architecture and the HIL approach.

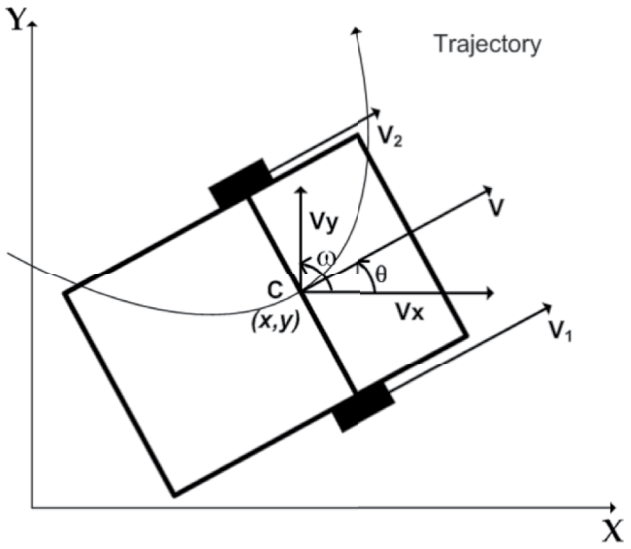


Fig. 1 Differential robot

Then, the dataset generation is shown in section 8. Lastly, conclusion and future work are presented in Sect. 9.

2 Kinematics

The applied mobile robot has a differential kinematics, presented in Fig. 1, and composed of two wheels, with their shaft passing by the same axle. Their movement is controlled by independently varying each wheel's velocity.

The mechanical structure of the differential robot, presented in Fig. 1, does not allow movements along the axle that cross wheel shafts. Considering that there is no lateral slip, each wheel velocity in the wheel's ground contact point is perpendicular to the axle along the wheel shaft.

In this case, $V_1(t)$ and $V_2(t)$ are the velocities measured in the contact point between the wheel and the ground. The robot's linear and angular velocities are given by Eqs. 1 and 2, where b represents the distance between the ground contact points.

$$v(t) = \frac{V_1(t) + V_2(t)}{2} \tag{1}$$

$$\omega(t) = \frac{V_1(t) - V_2(t)}{b} \tag{2}$$

Considering the non-slipped situation, the differential robot kinematics can be described by the following equations for a global localization reference system.

$$\dot{x} = v(t)cos(\theta(t)) \tag{3}$$

$$\dot{y} = v(t)sin(\theta(t)) \tag{4}$$

$$\dot{\theta} = \omega(t) \tag{5}$$

3 Mobile robot prototype

The mobile robot prototype, presented in Fig. 2, has a differential kinematics and was designed to be able to follow a line, follow a wall, and avoid obstacles. The mechanical design was developed by applying 3D printing technology. It is composed of a set of parts, including a structure for two wheels, a chassis to load the electronic components, and additional parts for attaching the sensors, actuators, and user interface, such as buttons. Four identical robots were assembled to be applied inside an educational context.

In order to achieve robustness and reliability for the robot, a custom printed circuit board (PCB) shield for Arduino

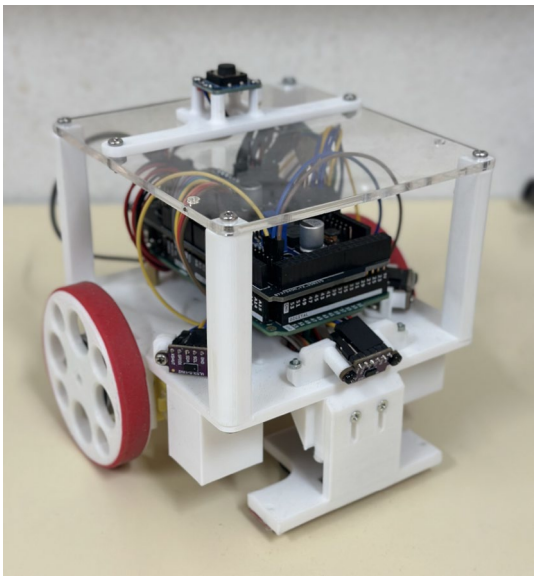


Fig. 2 Assembled mobile robot

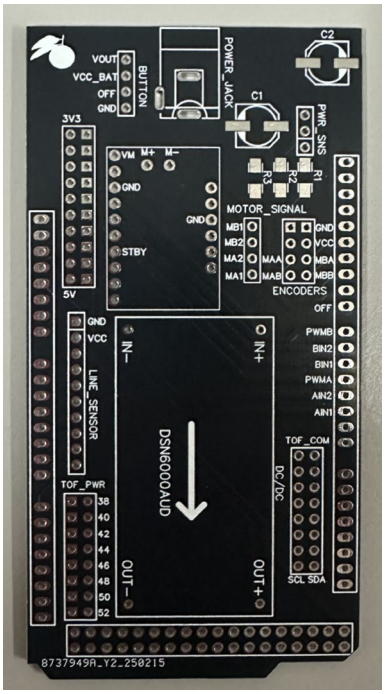


Fig. 3 Custom PCB shield for Arduino

Mega was designed. The custom PCB, presented in Fig. 3, integrates the robot components, like motor drivers, power regulation, and sensor connectors, into a single board, minimizing wiring complexity and consequently improving the electrical stability. Therefore, with this shield, better performance in real-world conditions, efficient operation, and resistance to environmental stresses were reached [31].

The electronic components were chosen considering the objective for the robot to follow a line, follow a wall, and

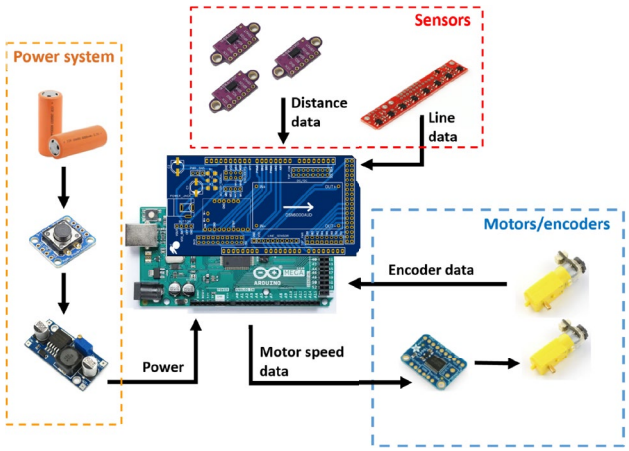


Fig. 4 Hardware components

avoid obstacles, having in mind that the robot must be cost-effective. The hardware components of the mobile robot, illustrated in Fig. 4, are composed of an Arduino Mega microcontroller responsible for processing the data, a QTR-8A line sensor to detect the line that the robot must follow, and three VL53L0X time-of-flight (ToF) sensors for obstacle avoidance and follow a wall. The power system includes two rechargeable 3.7 V Lithium Ion batteries, a step-down regulator, and an auto power-off function in order to prevent battery damage. Two DC motors with built-in encoders are used for locomotion, allowing a closed-loop speed control and odometry estimation. A FIT0450 drive board is also applied to control the motors.

4 Control system design

The goal of the robot is to follow a line or a wall; for that purpose, it is necessary to maintain the highest possible linear velocity with an angular velocity that is proportional to the path curvature. With the reflectance sensor, also known as a line sensor, it is possible to know the robot's posture in relation to the line at each sample time. In this way, the path curvature is known as being proportional to the line distance to the sensor center (*error*).

The robot's linear velocity can be given by the following Eq. 6:

$$v = K_1 - K_3 \cdot |error| \tag{6}$$

where K_1 indicates the top linear velocity and K_3 controls the velocity decrease as a function of the path curvature.

The robot's angular velocity can be given by the following Eq. 7:

$$\omega = K_2 \cdot error \tag{7}$$

where K_2 controls the angular velocity as a function of the curvature path.

The robot's movement is controlled by varying each wheel's velocity independently; in this way, the velocity for each wheel can be given by Eqs. 8 and 9, having in mind Eqs. 1 and 2.

$$V_1 = v + \frac{\omega b}{2} \tag{8}$$

$$V_2 = v - \frac{\omega b}{2} \tag{9}$$

From Eqs. 6, 7 and 8, Eq. 10 can be obtained:

$$V_1 = K_1 + \frac{K_2 \cdot error \cdot b}{2} - K_3 \cdot |error| \tag{10}$$

From Eqs. 6, 7 and 9, Eq. 11 can be obtained:

$$V_2 = K_1 - \frac{K_2 \cdot error \cdot b}{2} - K_3 \cdot |error| \tag{11}$$

4.1 Algorithm for line error calculation

The algorithm created for the robot to follow a line is based on calculating the relative error in relation to the line center. The module QTR-8A Reflectance Sensor was applied to detect the black line, it is composed of eight infrared LED/phototransistors, being able to provide different reflections based on the surface color [32].

The sensor outputs are composed of analog values that vary according to the surface color. A threshold was defined to detect the black line: values below this threshold indicate detection (active), while values above indicate no detection (inactive). Therefore, the sensor is treated as a digital device rather than an analog one. A weighted average is then calculated based on the positions of the active phototransistors that detect the black line.

Considering the eight phototransistors in the line sensor, the robot has eight measurement positions varying from − 3.5 to + 3.5, as illustrated in Fig. 5. In order for the robot to move over the line aligned with the line center, it is necessary converge to an error close to zero.

The approach is based on the following situation: if the robot is aligned with the line center, then the phototransistors 3 and 4 are active, indicating an error value equal to zero. If the robot is misaligned to the right of the robot, only phototransistor 4 is active, indicating an error value of + 0.5. Therefore, the algorithm for line center calculation is independent of line width; it is adaptable for any width.

For this methodology, a weighted average was applied to determine the error in relation of line center, presented in the Eq. 12, in which the weight for each sensor is represented by w_i, x_i represents each sensor value that can be 1 for active or 0 for not active, the sum of this multiplication is then divided by the sum of the number of active sensors n_i.

$$line\_error = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} n_i} \tag{12}$$

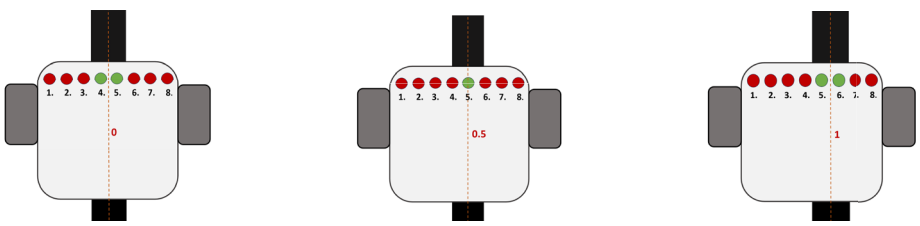
4.2 Algorithm for wall error calculation

The algorithm for wall following is similar to the one applied for line following, however its input is the error in relation to a predefined distance from the wall. In this context, if the error increases positively compared with this target distance, it indicates that the robot is moving away from the wall. Therefore, the robot should move closer to the wall. On the other hand, if the error increases negatively, the situation is the opposite: the robot is approaching the wall. Therefore, the robot must move far from the wall.

The diagram presented in Fig. 6 illustrates the trigonometric approach for this situation. The data provided by the right or left sensor is represented by *measured_distance*, which is a distance measurement not perpendicular to the wall but with 45º of rotation due to the position of the sensors in relation to the robot center.

Applying trigonometric calculations, the real distance from the robot to the wall is represented by *actual_ distance* and presented in Eq. 13. Considering a *target_distance* from the robot to the wall of 0.12 m, the error in relation to the wall can be expressed by Eq. 14. Therefore, the *wall_error* variable represents the error of robot position in relation to the target distance from the wall, and it must be included in the controller Eqs. 10 and 11, presented in the previous section. Normalization and saturation processes were also

Fig. 5 Line error approach



(a) Error = 0 (b) Error = 0.5 (c) Error = 1

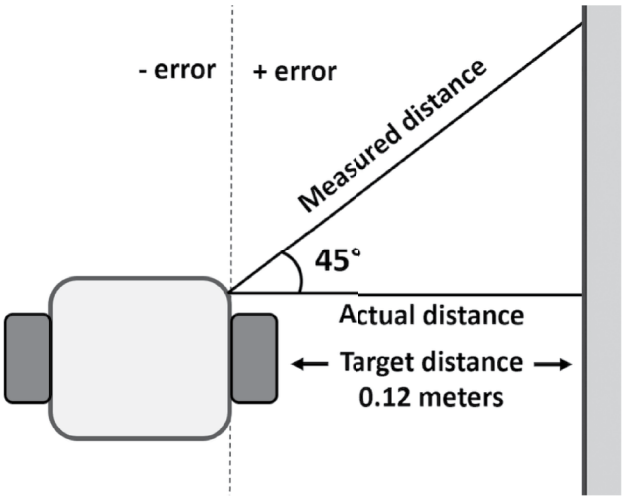


Fig. 6 Wall error approach

applied to these values in order to make them coherent with the same range of line error. This way, the controller can be the same for both situations: line following and wall following.

$$actual\_distance = cos(45^{\circ}) \cdot measured\_distance \quad (13)$$

$$wall\_error = target\_distance - actual\_distance \quad (14)$$

4.3 Obstacle avoidance approach

For obstacle avoidance, several approaches were applied, such as the information of ToF sensors, mainly of the front sensor, to change the state machine to perform other actions, such as avoidance. Odometry information can also be used for the robot to navigate without the measurements from a wall or a line support. The disadvantage of odometry calculation is that the error increases over time.

In addition, splines were applied for the obstacle avoidance, which are predefined curves that the robot executes around the obstacle. The data from the lateral sensors were applied to monitor when the obstacle was out of view and return to the standard state of following a line or a wall. Splines are a good option for generating continuous and different trajectories for obstacle avoidance, providing soft and controlled transitions between different parts of the path and guaranteeing stability in the robot's movement. These curves consist of multiple points, including control points that define the shape of the curve [33].

Several deviation trajectories were pre-calculated to avoid abrupt movements and adjust for different situations, for example, to avoid obstacles in a straight line or in a path curvature. The size of the obstacle and its position in relation to the robot were also taken into consideration. In this case, the pre-calculated splines had 3200 evenly spaced points and were stored in arrays of points, allowing the robot to execute each trajectory according to the situation.

Figure 7 presents some deviation trajectories created with different control points to be applied in various situations of obstacle avoidance, depending on the laterality of the obstacle (left or right), its proximity, and its position in the path. The X and Y axes of the graphics represent the X and Y positions, in millimeters, that the robot must move to. Figure 7a shows a trajectory used to avoid obstacles in a straight line, while Fig. 7b and c were applied in the situation of avoiding obstacles in a path curvature, from the outside and inside of the curve, respectively.

5 Sensors and actuators modeling

Modeling of sensors and actuators plays a crucial role in simulation and control system design, as they offer an accurate and structured way to represent complex systems. These models enable the analysis and prediction of system behavior under varying conditions by capturing the dynamic relationship between input and output signals within a defined state space [26, 34]. Therefore, this section presents the modeling process of the ToF sensor and the DC motors of the mobile robot, aiming to support the development of a simulation that closely reflects real-world behavior and enables the generation of accurate datasets.

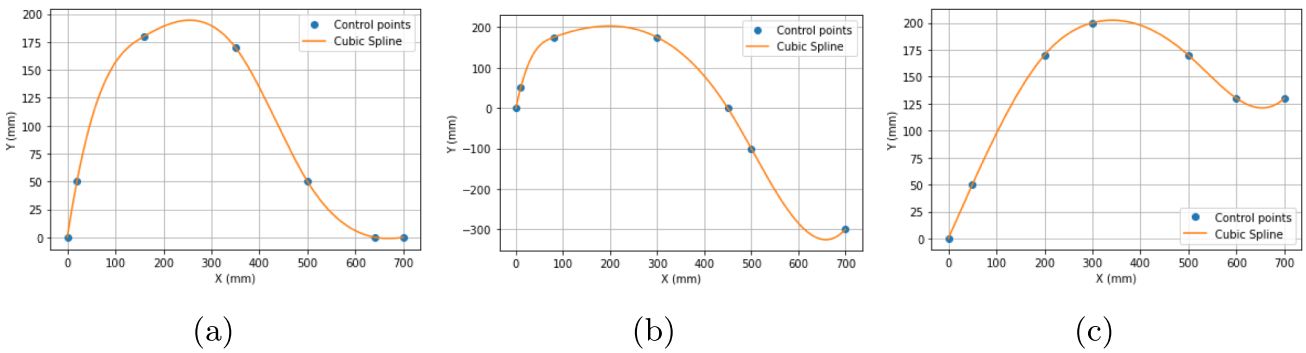


Fig. 7 Obstacle avoidance approaches **a** Straight line **b** Outside the path curvature and **c** Inside the path curve

5.1 Motor modeling

Representing the dynamic behavior of DC motors within the SimTwo simulator, a model was developed based on the fundamental physical principles governing this type of motor. Initially, the motor's operation is characterized using an electromechanical parameter model, as illustrated in Fig. 8.

The electrical characteristics of the system are represented by a series of three lumped elements: a resistor with resistance R, an inductor with inductance L, and a voltage source accounting for the back electromotive force produced by the rotor's motion. On the mechanical side, the model incorporates the rotor's moment of inertia J, a viscous friction coefficient B and a static friction coefficient T_Q. Applying Kirchhoff's voltage law, Eq. 15 was obtained, which represents the electrical behavior and the relation between input voltage $u(t)$ and electrical current $i(t)$.

$$u(t) = R \cdot i(t) + L \cdot \frac{di(t)}{dt} + e_v(t) \tag{15}$$

The counter-electromotive force is linearly proportional to the rotor speed $w(t)$, being K_e the proportionality constant and presented in Eq. 16.

$$e_v(t) = K_e \cdot w(t) \tag{16}$$

The mechanical behavior can be defined by Newton's second law, presented in Eq. 17, where the motor torque is represented by T_M that is a linear function of the electric current $i(t)$, defined by the relation $T_M = K_M \cdot i(t)$, being K_M a positive constant. The braking torque T_B, which results from friction, is proportional to the rotor's angular velocity $w(t)$ and given by $T_B = B \cdot w(t)$, being B the viscous friction coefficient. Additionally, a constant torque T_Q for the static friction is included to account for other unmodeled effects. Performing the substitutions, Eq. 18 was obtained.

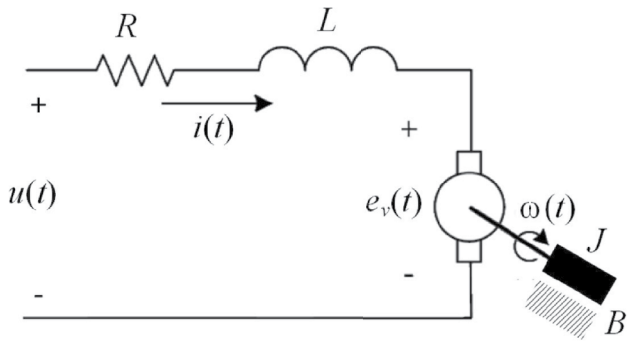


Fig. 8 Permanent magnet DC motor parameters model

$$T_M - T_B - T_Q = J \cdot \frac{d \cdot w(t)}{dt} \tag{17}$$

$$K_M \cdot i(t) - B \cdot w(t) - T_Q = J \cdot \frac{d \cdot w(t)}{dt} \tag{18}$$

From Eqs. 15 and 18, the second-order differential Eq. 19 was obtained. Then, neglecting the effect of the inductance and assuming $K_e = K_M = K$, a first-order differential Eq. 20 is obtained.

$$\begin{aligned} &\frac{LJ}{K_M}\frac{d^2\omega(t)}{dt^2} + \left(\frac{RJ}{K_M} + \frac{LB}{K_M}\right)\frac{d\omega(t)}{dt} \\ &\quad + \left(\frac{RB}{K_M} + K_e\right)\omega(t) + \frac{RT_Q}{K_M} = u(t) \end{aligned} \tag{19}$$

$$\frac{RJ}{K}\frac{d\omega(t)}{dt} + \left(\frac{RB}{K} + K\right)\omega(t) + \frac{RT_Q}{K} = u(t) \tag{20}$$

Therefore, it is necessary to determine the parameters R, B, K and T_Q, being them:

- R, the resistance of the motor winding (Ω)
- K—the electromechanical constant of the motor
- B—the coefficient of viscous friction
- T_q is a torque constant

In this context, considering the motor at a steady state, tests were conducted in which the electric current and rotational speed of the motor were measured for different voltage values, presented in Table 1.

Figure 9 illustrates the setup created for these tests. The data was acquired at a frequency of 25 Hz, being the setup composed of an ESP32 microcontroller, a 6 V DC motor with a built-in encoder, a motor drive TB6612, a current sensor INA219, and a computer. The microcontroller ESP32 is responsible for data processing and sending information to a computer through serial communication. The 6 V DC motor with a built-in encoder sends encoder data for esp32 in order to calculate the velocity once there is a closed-loop system. The ESP32 also controls the driver board in order to control the motor. A current sensor, INA279, is connected through the drive board and the motor terminals to measure the electric current.

Taking into account the steady-state motor behavior, the following system of equations was considered for the motor parameters estimation, in which the least squares method (LSM) was applied to obtain the best approximation. The first equation of the system represents the electrical behavior of the DC motor and the second equation represents the mechanical behavior.

Table 1 Steady-state current and angular velocity for different motor voltages

u (V)	I (A)	w (rad/s)
0.00	0.000000	0.00000
0.25	0.072723	0.00000
0.50	0.072155	0.87457
0.75	0.079350	1.61743
1.00	0.085538	2.42328
1.25	0.091426	3.24140
1.50	0.094087	3.96708
1.75	0.103298	4.76475
2.00	0.109114	5.45525
2.25	0.113620	6.23410
2.50	0.117927	7.11685
2.75	0.123966	7.80653
3.00	0.131561	8.55920
3.25	0.137696	9.20634
3.50	0.144181	9.99828
3.75	0.148593	10.75750
4.00	0.151752	11.48560
4.25	0.158205	12.32090
4.50	0.166269	12.89280
4.75	0.172286	13.50970
5.00	0.178680	14.31800

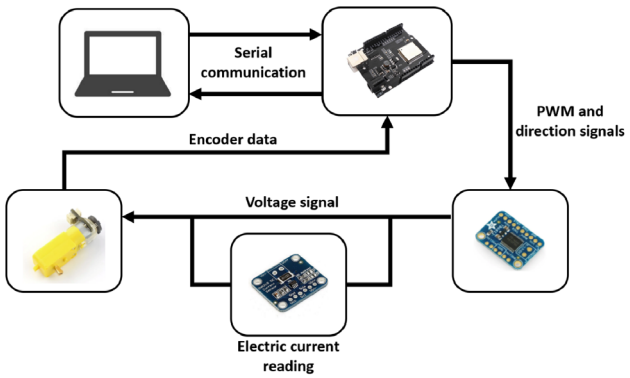


Fig. 9 Experimental setup for motor modeling

$$\begin{cases} u(t) = Ri(t) + K\omega(t) \\ 0 = Ki(t) - B\omega(t) - T_q \end{cases} \tag{21}$$

where:

- $u(t)$ is the voltage applied to the motor (V)
- $i(t)$ is the electric current (A)
- $\omega(t)$ is the angular velocity (rad/s)
- R is the resistance of the motor winding (Ω)
- K is the electromechanical constant of the motor
- B is the coefficient of viscous friction
- T_q is a torque constant

Table 2 DC Motor estimated parameters

Parameters	Values
R	3.03
K	0.31
B	0.002
Tq	0.02

Based on the acquired data presented in Table 1, firstly, the parameters R and K of Eq. 21 were estimated, applying the LSM method by solving the following Eq. 22:

$$\boldsymbol{\theta} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{u} \tag{22}$$

where $\mathbf{H}$ is the matrix of regressor vectors, $\mathbf{u}$ is the vector associated with the voltage values, and $\theta = [R \quad K]^T$. The same procedure was carried out for the second Equation represented in 21 in order to estimate B and T_Q. The results for the estimated parameters are presented in Table 2. Figure 10 shows the real and modeled motor steady-state response data, in which it can be concluded that the model shows a good fit to the experimental data.

5.2 ToF sensor modeling

The ToF sensor is a ranging sensor manufactured by STMicroelectronics very popular due to its high accuracy, easy-of-use, and low cost. Operating with an infrared signal of 940 nm, capable of reaching distances up to 2 m and with a narrow field of view of 25°. The sensor is based on the ToF technology [35], illustrated in Fig. 11, in which a modulated infrared signal is sent to a target and the reflected signal is captured by a receptor, converting photonic energy to electrical current. Based on the reflection time of the signal, the distance is calculated and presented as an output of the sensor [36–38]. This sensor can work in four ranging modes: the default mode, high speed, high accuracy, and long range. The default mode was chosen to be enough for the proposed application because provides a tradeoff between reactiveness and accuracy; this operating mode contains a range timing of about 30 ms [35, 38].

5.2.1 Setup for acquiring data

For the modeling process of the ToF sensor, a setup was created to acquire data, using a collaborative robot as a manipulator in order to obtain movements with accuracy. The used robot was a collaborative robot from Kassow Robots, which has 7 degrees of freedom, a payload of 10 Kg, and a reach of approximately 1 m [39, 40]. The ToF sensor modeling details have already been presented in [27, 38, 41].

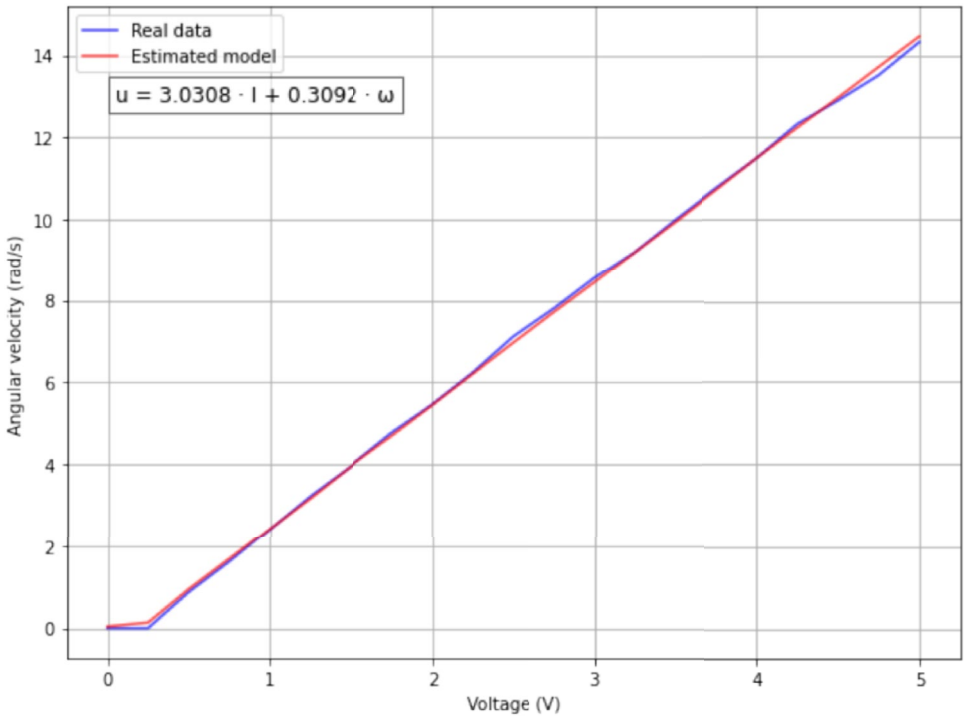


Fig. 10 Motor steady state response data (modeled vs. real)

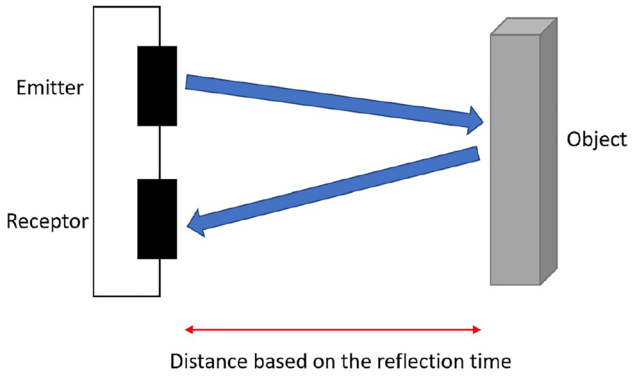


Fig. 11 Time-of-flight principle illustration

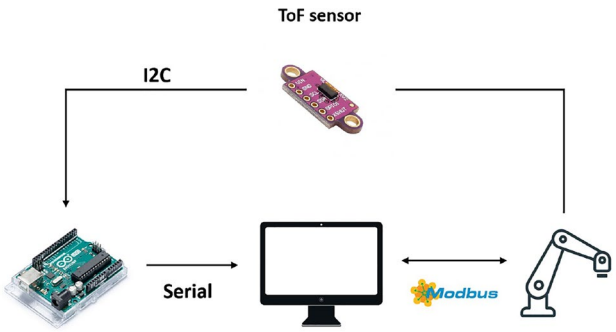


Fig. 12 System data flow for ToF sensor modeling

The system communication, illustrated in Fig. 12, involves the ToF sensor fixed to the flange of the collaborative robot with a 3D printed support Fig. 13a. The sensor sends data via I2C to an Arduino, which transmits the measurements to the computer via serial communication. The computer exchanges data with the robot via the Modbus over Ethernet protocol, chosen for its compatibility and available libraries. The system operates with three parallel processes: the microcontroller sends measurements at 30 Hz, the computer performs control and processing, and the robot executes the trajectories and sends feedback [27, 38].

Five experiments, illustrated in Fig. 13, were performed to collect data from the ToF sensor. In the first one, Fig. 13b, the behavior of the sensor was analyzed when faced with different shades of paper (white, black and two shades of gray) and heights ranging from 0 to 1.2 m in 1 mm steps. In the second test, Fig. 13c, the influence of different materials such as metal, wood, glass, paper and plastic with distances ranging from 61 to 900 mm was evaluated. The third experiment, Fig. 13d, consisted of the horizontal movement of the robot over a step, using a white surface, to observe the sensor's response to a sudden change in distance. In the fourth experiment, Fig. 13e, the robot adjusted the sensor's tilt angle while maintaining the distance to a white table, moving it laterally and forwards/backwards, in order to analyze the field of view and the effect of orientation. Finally, in the fifth experiment Fig. 13f, an infrared-sensitive PiCamera was used to capture images of the ToF sensor's field of view and compare them with the datasheet data.

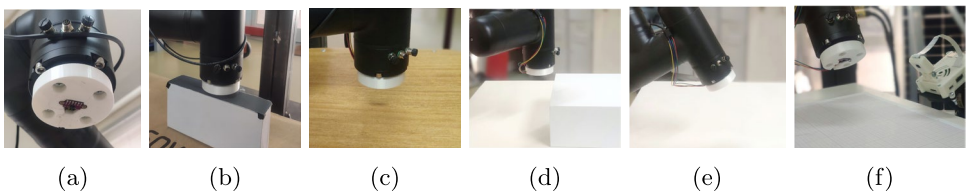


Fig. 13 Complete setup to acquire data. **a** ToF sensor attached to the robot flange **b** Manipulator moving the sensor vertically in different colors tones **c** Manipulator moving the sensor horizontally over a step **d** Manipulator moving the sensor in different angles **e** Manipulator moving the sensor vertically in different surfaces types—in this picture wood **f** Camera obtaining images from sensor field of view

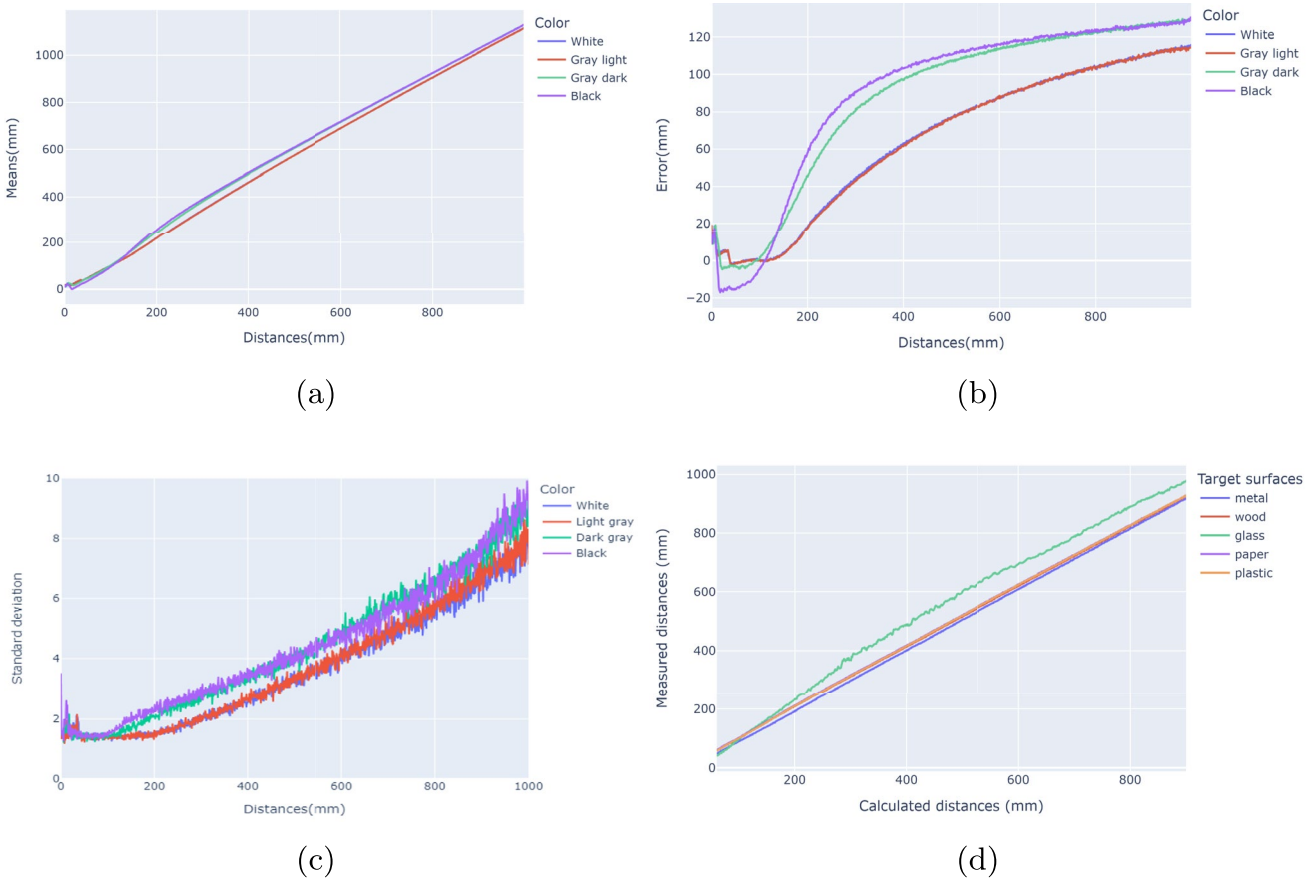


Fig. 14 **a** Mean **b** Error and **c** Standard deviation for different colors tones **d** Mean for different surfaces

Table 3 Mean and standard deviation for distances at 100 mm in different colors tones

Colors	White	Light gray	Dark gray	Black
Mean (mm)	100.25	100.21	101.24	95.63
Std	0.84	0.82	0.79	0.80

5.2.2 ToF sensor modeling results

From the data obtained in the experiments, the mean, error and standard deviation for each color tone were calculated and presented in Fig. 14a–c, respectively, as well as the results at 100 mm were presented in Table 3. Before 40 mm and after 1 m, the measurements were considered inconsistent because there were disturbances and failures in the values measured by the sensor. It is possible to conclude that for white and light gray, the average values are close to the theoretical distance of 100 mm. Dark gray and black color values disperse, which is expected due to the low reflection level of these colors. The white color presented the biggest standard deviation, providing distributed values for this color, and dark gray was the color that presented the smallest standard deviation, which indicated that the data were not evenly distributed in relation to others [38].

The mean curve approximates a linear curve, and according to the distance increase, the error increases too, which are both expected. It can be concluded that the sensor behavior was better for short distances, around 40 to 180 mm, where the mean curve is totally linear, as it can be seen in Fig. 14a,

Table 4 Mean, error and standard deviation for distances at 100 mm in different surfaces

Surfaces	Metal	Wood	Glass	Paper	Plastic
Mean (mm)	89.06	99.88	96.58	101.42	99.7
Error (mm)	± 10.94	± 0.12	± 3.42	± 1.42	± 0.29
Std	1.28	1.42	1.77	1.45	1.31

the error is close to zero, as shown in Fig. 14b, and the standard deviation remained constant. Another point observed is the difference between the curves for light colors (white and light gray) and dark colors (dark gray and black), which is caused by the white color having a reflection level higher than the others [38].

For the second experiment, Table 4 shows the results obtained at a distance of 100 mm, highlighting that wood, paper and plastic presented averages closer to the theoretical value and smaller errors. Metal and glass had greater dispersion and standard deviation, with glass being the most unstable. Figure 14d illustrates the averages as a function of the theoretical distances, showing that the curves are almost linear, except in the case of glass, which presented greater errors due to low reflection and high refraction. Aluminum, being very shiny, generated unwanted reflections. It can be concluded that the reflectance of the material directly influences the measurements: surfaces such as glass and metal compromise accuracy, while paper, wood and plastic guarantee more stable and accurate results.

Curves for standard deviation and error were estimated in order to obtain stochastic modeling for the ToF sensor, providing relevant data that can be applied in a simulation environment. This approach enhances the sensor's usability and provides valuable data for applications involving sensor fusion.

The standard deviation estimation was obtained by minimizing the absolute error between the actual data and a third-order polynomial curve, as shown in Fig. 15. The data from white surfaces were used as a reference in this process since this color yielded the most reliable performance. A minimum distance of 0.04 m was selected because measurements below this threshold tend to be unreliable. Between 0.04 and 0.18 m, the standard deviation remains nearly constant, with an estimated value of 1.41. Beyond this range, the standard deviation follows a third-degree polynomial behavior. The two mathematical expressions used are shown in Eq. 23, where x denotes the distance in meters and y represents the standard deviation in millimeters.

$$\begin{cases} y(x) = 1.41 \quad \text{when} \quad 0.18m > x > 0.04m \\ y(x) = 1.15x^3 + 2.51x^2 + 4.88x + 1.40 \quad \text{when} \quad x > 0.18m \end{cases} \tag{23}$$

For estimating the measurement error, a logarithmic curve was fitted using a quadratic error minimization approach. The resulting model is presented in Eq. 24 and visually depicted in Fig. 16. The curve starts at 0.18 m, as the error values below this distance are minimal and can be reasonably approximated as zero. In the equation, x corresponds to the distance in meters, while y represents the measurement error in millimeters.

$$y(x) = 59,77 \ln(x) - 116.86 \tag{24}$$

In the third experiment, the manipulator moved the sensor horizontally over a white paper box with a height of 106 mm, starting from 0 to 300 mm (1 mm steps) and at six different distances from the target (ranging from 50 to 300 mm). The results, presented in Fig. 17, show that the measurements

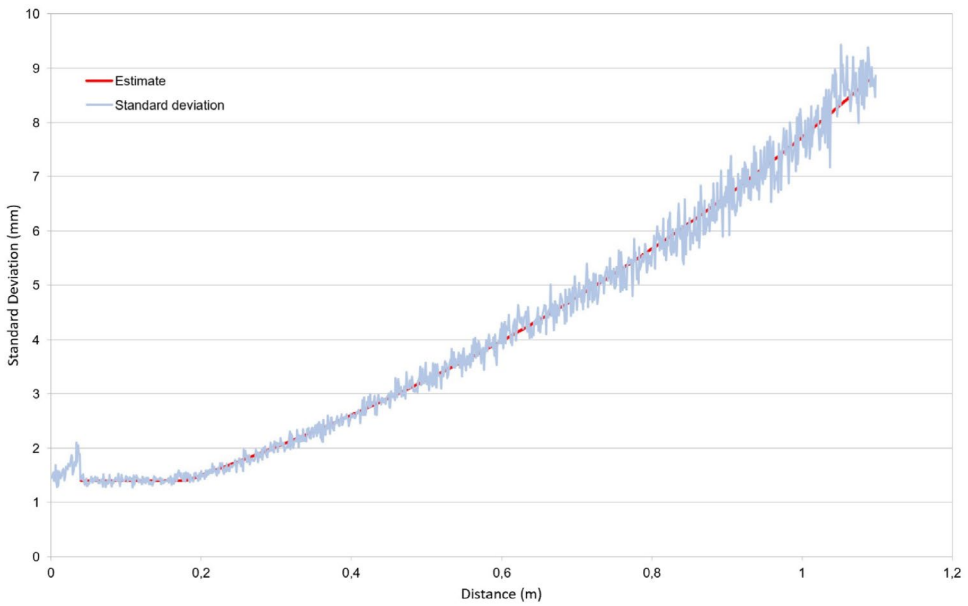
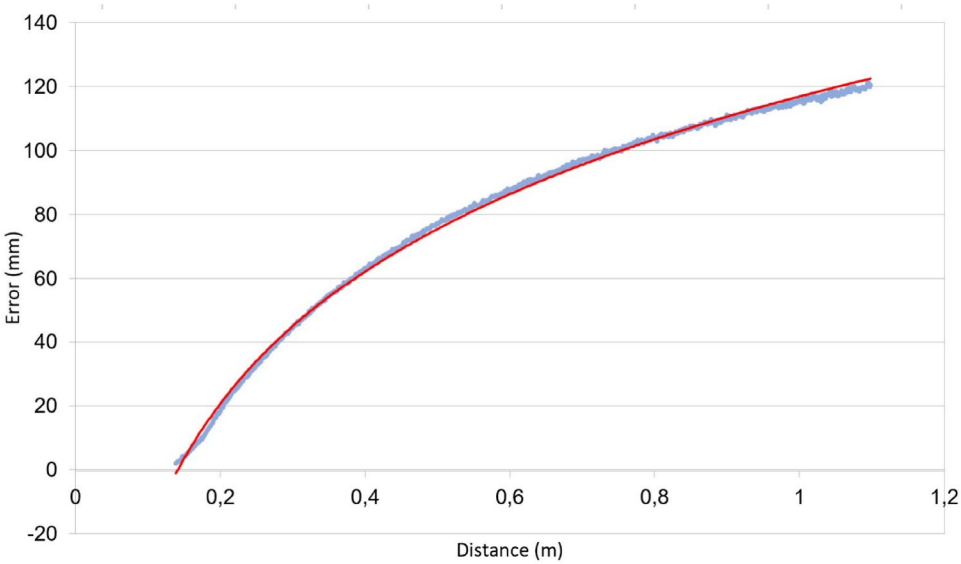


Fig. 15 System data flow

Fig. 16 System data flow



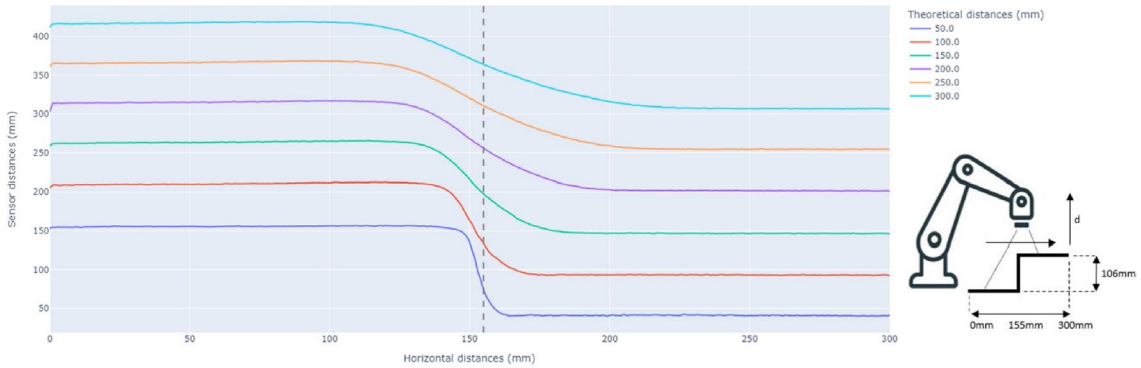


Fig. 17 System data flow

before the step and after the step, once adjusted for the box height, were close to the theoretical values, confirming the consistency of the data. It was observed that the greater the distance from the sensor, the smoother the curves due to the larger field of view that captures points outside the step. The increase in error proportional to the distance demonstrates good sensor response even when faced with abrupt height variation [38].

In the fourth experiment, the manipulator tilted the sensor at different angles (from − 25° to +25°, 2° steps) and distances from the target (50–300 mm), maintaining the same distance in relation to the white surface, as presented in Fig. 18. The measurements, at 0°, were close to the theoretical values, but when tilting the sensor, errors arose due to misalignment between transmitter and receiver, affecting the capture of the reflected signal. When moving the sensor laterally, the curves became more symmetrical, while when moving back and forth there was a greater inclination on one side, probably caused by the physical position of the emitters and receivers or manufacturing asymmetries, despite the leveling of the support having been verified with a spirit level.

In the last experiment, the sensor's field of view was analyzed using an infrared-sensitive camera, comparing the measured diameters with the theoretical values calculated from the 25° angle specified in the datasheet. As the sensor moves away, its field of view increases, which can generate more errors in the measurements. The captured images presented in Fig. 19 showed results consistent with the theoretical values, confirming the accuracy of the calculations, as detailed in [27]. The tests were limited to 150 mm, because at greater distances the projected circle becomes too wide to be measured clearly. Captures were also taken with the sensor tilted, maintaining the distance, and with the ambient lighting turned off to better observe the deformation of the field of view, as detailed in [27].

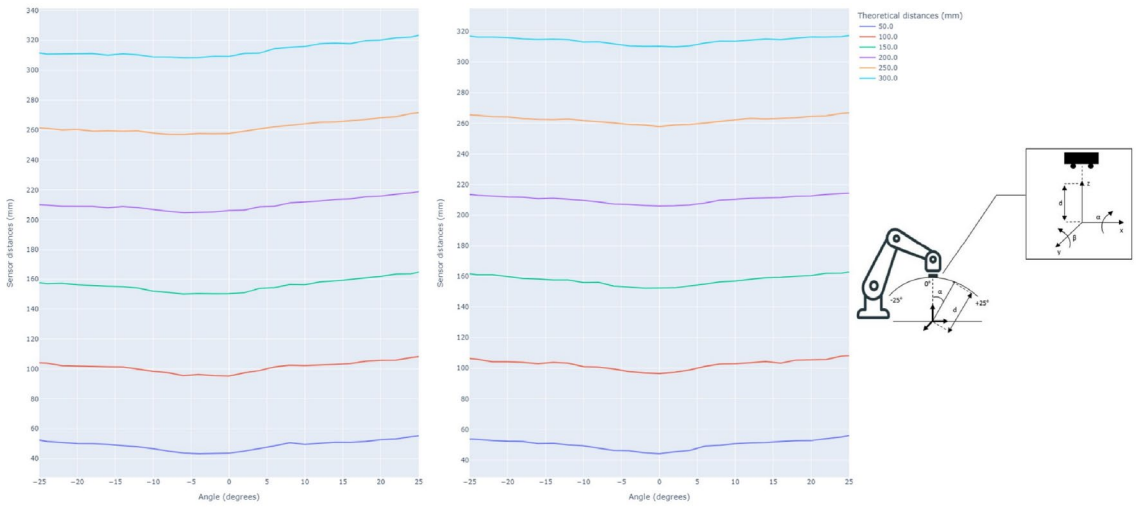


Fig. 18 System data flow

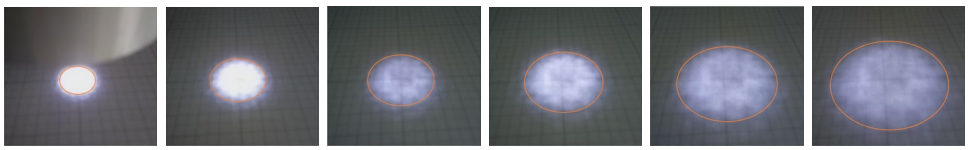


Fig. 19 Results of the field of view according to different heights. **a** 50 mm of height **b** 70 mm of height **c** 90 mm of height **d** 100 mm of height **e** 130 mm of height **f** 150 mm of height

6 Simulation environment

For the development of realistic simulation, an open source robotics simulator called SimTwo [42] was applied, which is based on the Open Dynamics Engine [43], an open source library designed for simulation of rigid body dynamics. One of the main advantages of using a simulator is the possibility of developing robot control applications without the need of having a physical robot [44–46].

SimTwo has been applied in an educational setting, assisting in developing robot control software even without access to real hardware. Additionally, this simulator offers a 3D environment that supports various types of robots, including omnidirectional wheeled robots, industrial robots, humanoids, and others, and it includes a wide range of components such as sensors and actuators, whose models can be added as inputs. The ability to incorporate real models of sensors, actuators, rigid-body parts, and robot joint configurations brings dynamic realism to SimTwo, enabling simulations that closely resemble real-world behavior. Furthermore, SimTwo also provides the possibility to send and receive data remotely because it includes several types of communications, making possible the application of, for example, techniques such as Hardware-in-the-loop, which will be discussed posteriorly [41, 45–48].

Before the creation of the virtual robot, the ToF sensor was simulated inside SimTwo in order to test if the model in the simulation reflects the characteristics of the real sensor.

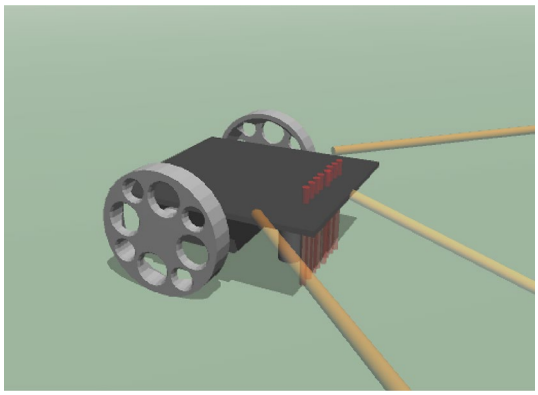


Fig. 20 Virtual robot

For this simulation, the sensor was integrated into the SimTwo environment using an array of 90 rays to approximate the dispersion of infrared signals across its field of view of 25°. The simulated output is obtained by computing a weighted average of the 90 ray measurements, after which noise and error models are applied to emulate real-world sensor behavior. Detailed information about this work can be found in [41]. The results closely matched those obtained from the real tests. This consistency indicates that the model developed for the ToF sensor accurately reflects real-world behavior.

Then, a virtual robot, presented in Fig. 20 with the same dimensions as the real robot, was created inside the simulation environment. The virtual model consists of solid

shapes, such as cuboids and cylinders, connected by joints like hinges, which allow movement along a defined axis. Taking into consideration the modeling of sensors and actuators presented in Sect. 5, the parameters obtained from the motors' modeling and also encoders data were included in the virtual robot in order to make the actuation of the robot close to reality. For the ToF sensor, the model obtained was included in the simulation, with all the specifications and equations from error and noise, making the sensors' data acquisition close to reality.

The scenario presented in Fig. 21 was created inside the simulation environment, considering the challenges of following a line, following a wall, and avoiding obstacles that were proposed inside a classroom context of master students. For the line purpose, two straight lines and two arcs were created and positioned in order to form a path inside the SimTwo, totalizing a length of 1.94 m. For the wall purpose, four solids representing the walls were positioned around the line, being two smaller walls with 0.8 m of lenght and two bigger walls with 1.5 m of length. In addition, an obstacle was also created with a dimension of $0.1 \times 0.1 \times 0.1$ m, and it could be positioned in different positions of the scenario, including on the line and close to the walls.

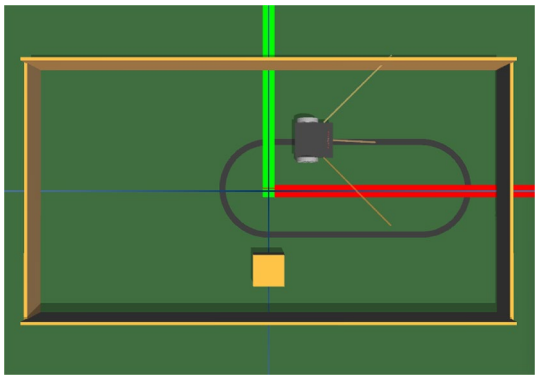


Fig. 21 Scenario created in SimTwo

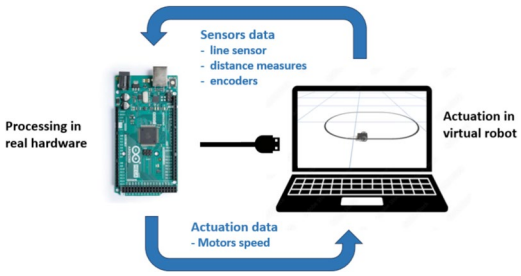


Fig. 22 Hardware-in-the-loop method

7 Software architecture

Hardware-in-the-loop (HIL) is a good approach to be applied inside an educational context, considering the situation in which students do not have access to a physical robot, for example. Applying this method inside classrooms allows students to evaluate and refine their control algorithms using real hardware processing conditions without risking damage to physical components. In addition, it enables testing and algorithm development by the students even when physical platforms are unavailable or already in use by other groups [28, 29, 44, 49, 50].

This method integrates real-time processing with simulated environments by establishing communication between a virtual robot in the simulator and the physical control hardware. Through the application of the HIL technique, the controller is able to operate under real hardware conditions while the sensors' data acquisition and actuation are carried out within the SimTwo simulator. As a result, students and developers can test and validate control algorithms in a realistic yet risk-free environment [28–30, 44, 49–51].

Figure 22 illustrates the HIL technique for this case, in which the simulator emulates sensor outputs, such as data from line sensors, distance measurements, and encoder values. These data are sent to the embedded system that processes this information and returns control signals, such as motor speeds. These signals drive the virtual robot, allowing for accurate simulation of physical behavior based on real-time decision-making.

The software architecture adopts two approaches. The first is real robot programming, where the microcontroller acquires, processes, and acts on data from physical sensors and encoders. The second uses the Hardware-in-the-Loop (HIL) method, where only the microcontroller is real, virtual sensors and actuators in a simulator provide data and receive commands, while the processing still runs on the microcontroller.

Figure 23 shows the flowchart of the software architecture, where the blue blocks indicate microcontroller tasks, green blocks the simulator tasks, and red blocks the parts the students must develop. Therefore, a library provides preprocessed data such as line sensor error, sensor values, distance, odometry (x, y, θ), motor setpoints and the controller. Students use this information to implement the control logic to transit between different operational states of navigation to overcome the proposed challenges such as line-following, wall-following and obstacle avoidance along the path.

8 Dataset generation

Aiming to contribute to the advances of localization techniques for mobile robots, this section describes the creation of a synthetic dataset openly available on [52], in order to

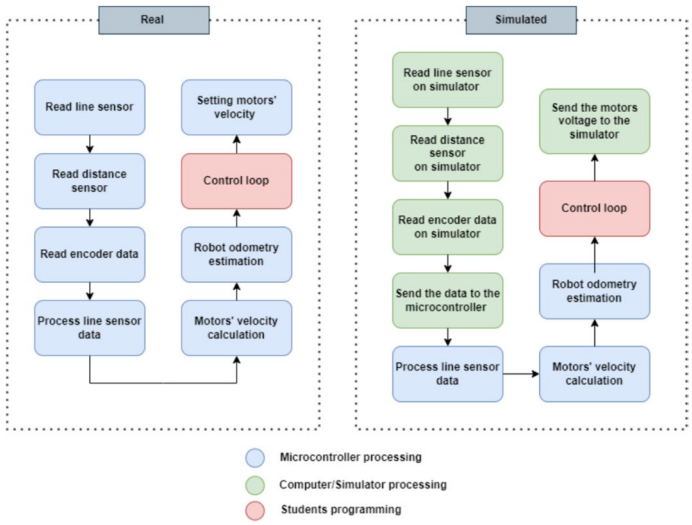


Fig. 23 Software architecture

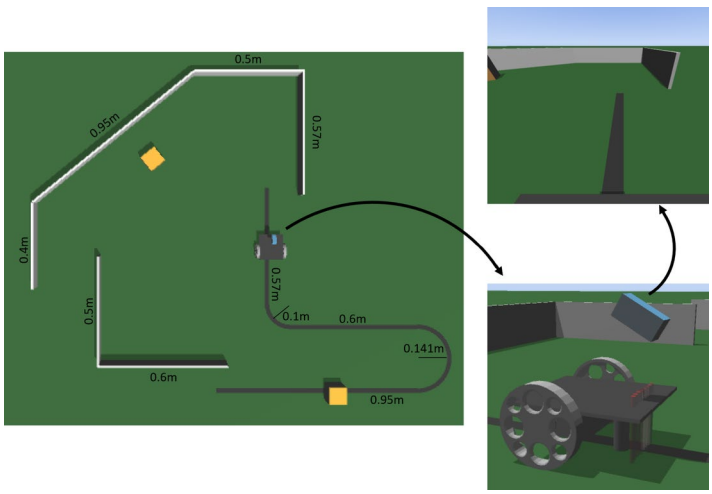


Fig. 24 Scenario and virtual robot applied for the dataset generation

facilitate reproducibility and to save cost and time. Making this dataset open-source, it is intended to support the scientific and educational community in the development and evaluation of robust and safe solutions for autonomous mobile robots.

8.1 Setup for data collection

The Synthetic Dataset for Mobile Robots Localization Algorithms (SyMoLo) was proposed and generated by a realistic simulation using the SimTwo simulator, already described in Sect. 6. The simulation allows the accurate control of variables, the generation of large volumes of data, and the replicability of experiments, being a valuable tool for testing and validating navigation and localization algorithms.

The same virtual robot already described in Sect. 6 and shown in Fig. 20 was applied for the dataset generation. A virtual camera was included above the robot in order to obtain images of the trajectory and scenario in front of the robot, as presented in Fig. 24. The inclusion of images of the path in the dataset is essential to provide visual information that assists in the localization and navigation of robots, allowing algorithms to learn to recognize environments, identify visual landmarks, and make more accurate decisions during movement. The camera had a focal length of 30 m and was positioned at 0.07 m of height from the robot base and 60° of inclination in relation to the axis perpendicular to the ground. The specific information about robot dimensions are available in the description document inside the dataset.

The SyMoLo dataset was gathered in a virtual scenario presented in Fig. 24, created inside the simulator SimTwo and covering the main tasks of the mobile robot: following a line, following a wall, and avoiding obstacles. In this context, the scenario includes walls in different positions and inclinations, lines with different curvatures, and obstacles along the route, exploring several challenges involving the exchange of the robot's states. The obstacles have a dimension of $0.08 \times 0.08 \times 0.12$ m and the height of all the walls

Table 5 Types of collected data

Data	Unit/format	Description
t	milliseconds	Timestamp at each data sample, since the start of the experiment
x	meters	Robot's position along the X-axis in global reference
y	meters	Robot's position along the Y-axis in global reference
theta	rad/s	Orientation of the robot
state	n/a	Current operational mode of the robot, such as line following, wall following, obstacle avoidance and others
v	m/s	Linear velocity of the robot
w	rad/s	Angular velocity of the robot
d_f	meters	Distance measured by the front ToF sensor
d_l	meters	Distance measured by the left ToF sensor
d_r	meters	Distance measured by the right ToF sensor
L0...L7	n/a	Readings from QTR-8A line sensor (array of 8 reflective phototransistors)
img	jpg	Image number assigned to this sample. Obs: if the number is 0, there is no image assigned to this sample

is 0.15 m. The other dimensions of the path are specified in Fig. 24.

Therefore, in this experimental scenario, the robot initially performs line following. Then, it must change to wall-following while simultaneously detecting and avoiding an obstacle. After that, the robot is required to switch to following a wall on the opposite side. Finally, it returns to the line following, during which it must again detect and avoid another obstacle.

The control system design described in Sect. 4 was implemented in the virtual robot's control loop, and data were collected using the Hardware-in-the-Loop (HIL) approach outlined in Sect. 7. For the obstacle avoidance task, the spline curve similar to that one shown in Fig. 7a was used to guide the robot to avoid obstacles on the left side while performing line following. In wall following mode, however, the side chosen for avoidance depends on which side the robot is following the wall and which direction is unobstructed, as determined by the sensor measurements.

8.2 Data collection methodolody

The SyMoLo dataset collected several types of data, presented in Table 5, that represent the state and the perception of the robot through the simulated trajectories. The quantitative data were captured at a sampling frequency of 20 Hz, therefore the timestamp between samples is around 50 ms. These data include the readings from the three ToF sensors

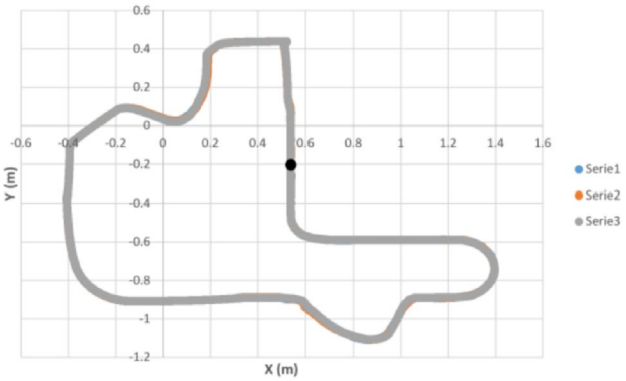


Fig. 25 Clockwise route

that provide real-time measurements of obstacles around the robot. Readings from the line sensor QTR-8A, are normalized reflectance values between light and dark runway surfaces, allowing to identify the relative position of the robot in relation to the black line on the floor. Other data are also included, such as the absolute position and orientation of the robot (ground truth) in the simulated environment, extracted from the simulator, linear and angular velocities of the robot during the path.

The states of the robot, that is, the operation modes are also collected in each sample. The robot has three main states: zero for line following, 1 for right wall following, 2 for left wall following, and 5 and 6 for obstacle avoidance by left and right side respectively. Then, there are other states related to specific movements, such as turn right and turn left, represented by states 4 and 3. There are also states 33 and 44 destinated for the situations when the robot gets lost while performing wall following, therefore the robot must move in the direction of the sensor that is in use for tracking.

Besides that, the dataset also includes images captured by a virtual frontal camera at a sampling frequency of 5 Hz (200 ms of timestamp). For each generated image, the corresponding robot coordinates are also provided in the simulated world, allowing direct associations between visual perception and localization. These data are temporally synchronized, enabling combined analysis and application in mapping, navigation, computer vision, and machine learning tasks.

The mobile robot performed two distinct paths, one in the clockwise direction (CW) and another in the counter-clockwise (CCW) direction. For each trajectory, three runs were conducted, totaling six experimental trials. Figures 25 and 26 presented the results of the paths performed by the virtual robot in the experimental scenario. The black point illustrates the starting point ($x = 0.55, y = -0.2, \theta = 90°$). The series specified in the legend of the figures correspond

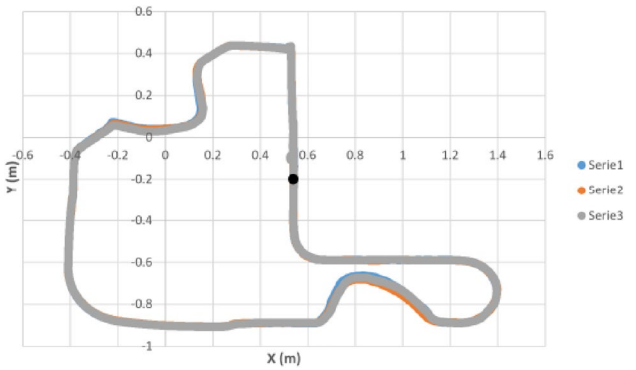
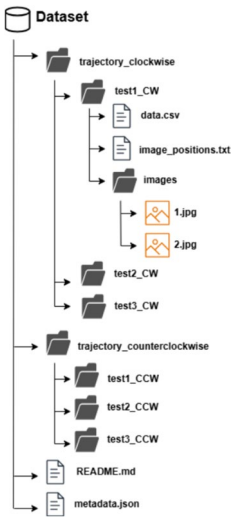


Fig. 26 Counterclockwise route

Fig. 27 Dataset structure



to the three runs. Both paths presented very similar trajectories, showing consistency in navigation behavior.

8.3 Dataset structure

The structure of the dataset is organized in folders and files as illustrated in Fig. 27. There are two folders destined for the two paths, one in the clockwise direction and the other in the counterclockwise direction. Within each trajectory folder, there are 3 more folders dedicated to each of the 3 tests carried out on the same route.

Inside each test folder, there is a *data.csv* file containing data collected from the sensors, robot positions, velocities, and associated images. There is also an *image_positions.txt* file that includes the position of the robot for each captured image. The JPEG images captured are stored in the *images* folder, and a selection of them is presented in Fig. 28 in order to assist in the understanding of the dataset's structure, usability and representative content.

The SyMoLo dataset also includes a markdown file with a detailed description of the dataset and a JSON file containing metadata information, such as the description of the robot, layout of the scenario, unities, simulator version, sampling frequency, and other data that is important for reproducibility.

9 Conclusion and future work

This article contributes to advancing and enhancing localization techniques for autonomous mobile robots, through the development of a synthetic dataset, generated by a realistic simulation of a mobile robot using the SimTwo environment inside an educational context. The dataset is publicly available, facilitating reproducibility and benchmarking, while reducing costs and complexity associated with real-world data collection. In addition, it enables researchers to train, test, and refine classical and emerging localization algorithms under customizable and controlled conditions.

The developed synthetic dataset combines quantitative data with visual information, providing a rich and structured source for training and validating algorithms of mobile robots localization. By simulating realistic scenarios with controlled variables, this dataset enables the testing of perception and control strategies in a reproducible and scalable way, making it particularly valuable for benchmarking performance under diverse conditions without the limitations imposed by physical hardware.

For the development of realistic simulation, the modeling of sensors and actuators contributes to inserting the realism and practical values in the simulation, therefore the resulting synthetic dataset includes noise, delays, and physical constraints also present in the real systems. Consequently, it is more suitable for developing and testing algorithms that are intended to operate reliably on actual robots.

The paper also addresses the development, control design, and prototyping of the physical mobile robot, allowing the validation of the realism and practical relevance of

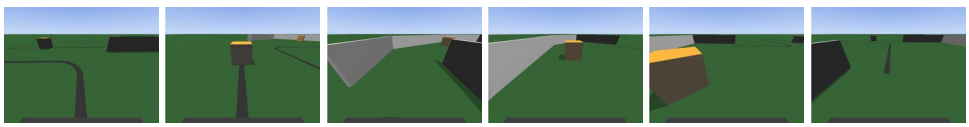


Fig. 28 Dataset images in the clockwise route **a** Following line **b** Following line close an obstacle **c** Following right wall **d** Following left wall **e** Avoiding obstacle **f** Following the left wall near the change to following line

the synthetic data, ensuring that the models and simulated scenarios are close to real-world conditions. Besides that, this approach increases the applicability of the dataset for researchers aiming to deploy their algorithms on real robotic platforms.

Therefore, the results of this work show that the simulated dataset can effectively replicate real-world conditions and behaviors, representing an effective strategy for educational and research contexts. In addition, to stimulating hands-on learning and the development of technical skills, this approach contributes to the creation of open and reproducible resources, which are essential for advancing teaching and research in mobile robotics.

As future work, it is planned to develop the cross-validation between real data from the physical robot platform and simulated data in order to improve the dataset. It is also intended the dataset enlargement, with more complex scenarios, dynamic environments, and different configurations of sensors. In addition, the development of localization, mapping, and navigation algorithms based on machine learning trained with the generated dataset is planned to evaluate their performance and robustness in different conditions. It is also planned to integrate the dataset with other popular simulation platforms such as ROS and Gazebo, expanding applicability.

Author contributions All authors made substantial contributions to the work reported in this manuscript and approved the final version for publication.

Funding This work was supported by FCT - Fundação para a Ciência eTecnologia, I.P. by PhD Grant 2023.01441.BD (https://doi.org/10.54499/2023.01441.BD) and2023.01113.BDANA (https://doi.org/10.54499/2023.01113.BDANA). This work was alsosupported by FCT - Fundação para a Ciência e Tecnologia, I.P. by projects: CeDRI,UID/05757/2025 (DOI: 10.54499/UID/05757/2025) and UID/PRR/05757/2025 (DOI:10.54499/UID/PRR/05757/2025); SusTEC, LA/P/0007/2020 (DOI: 10.54499/LA/P/0007/2020).

Data availability The dataset generated in this study is publicly available in the Zenodo repository, at the following link: https://doi.org/10.5281/zenodo.15806257.

Declarations

Conflict of interest The authors declare no conflict of interest.

Ethics approval and consent to participate Not applicable.

Consent for publication Not applicable.

References

1. Wijayathunga, L., Rassau, A., Chai, D.: Challenges and solutions for autonomous ground robot scene understanding and navigation in unstructured outdoor environments: a review. Appl. Sci. **13**(17), 9877 (2023)
2. Brancalião, L., Gonçalves, J., Conde, M.Á., Costa, P.: Systematic mapping literature review of mobile robotics competitions. Sensors **22**(6), 2160 (2022)
3. Al Mahmud, S., Kamarulariffin, A., Ibrahim, A.M., Mohideen, A.J.H.: Advancements and challenges in mobile robot navigation: a comprehensive review of algorithms and potential for self-learning approaches. J. Intell. Robot. Syst. **110**(3), 120 (2024)
4. Rybczak, M., Popowniak, N., Lazarowska, A.: A survey of machine learning approaches for mobile robot control. Robotics **13**(1), 12 (2024). https://doi.org/10.3390/robotics13010012
5. Yin, Y., Chen, Z., Liu, G., Yin, J., Guo, J.: Autonomous navigation of mobile robots in unknown environments using off-policy reinforcement learning with curriculum learning. Expert Syst. Appl. **247**, 123202 (2024). https://doi.org/10.1016/j.eswa.2024.123202
6. Hamid, T., Rasoul, H.: Deep reinforcement learning with enhanced PPO for safe mobile robot navigation (2024)
7. Ou, Y., Cai, Y., Sun, Y., Qin, T.: Autonomous navigation by mobile robot with sensor fusion based on deep reinforcement learning. Sensors **24**(12), 3895 (2024). https://doi.org/10.3390/s24123895
8. Li, Y.: Multimodal visual image processing of mobile robot in unstructured environment based on semi-supervised multimodal deep network. J. Ambient. Intell. Humaniz. Comput. **11**(12), 6349–6359 (2020)
9. Miller, A., Yu, F., Brauckmann, M., Farshidian, F.: High-performance reinforcement learning on spot: Optimizing simulation parameters with distributional measures (2025). arXiv preprint arXiv:2504.17857
10. Schuster, A., Hagmanns, R., Sonji, I., Löcklin, A., Petereit, J., Ebert, C., Weyrich, M.: Synthetic data generation for the continuous development and testing of autonomous construction machinery. Automatisierungstechnik **71**(11), 953–968 (2023)
11. Tobin, J.P.: Real-World Robotic Perception and Control Using Synthetic Data. University of California, Berkeley (2019)
12. Hwang, H., Adhikari, K., Shodhaka, S., Kim, D.: Synthetic data augmentation for robotic mobility aids to support blind and low vision people (2024). arXiv preprint arXiv:2409.11164
13. Choudhary, T.: The role of synthetic data in robotics: accelerating development and innovation. Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol. **11**, 2991–2998 (2025). https://doi.org/10.32628/CSEIT251112318
14. Schieber, H., Demir, K.C., Kleinbeck, C., Yang, S.H., Roth, D.: Indoor synthetic data generation: a systematic review. Comput. Vis. Image Underst. **240**, 103907 (2024)
15. Xu, Z., Nair, A., Xiao, X., Stone, P.: Learning real-world autonomous navigation by self-supervised environment synthesis (2022). arXiv preprint arXiv:2210.04852
16. Scucchia, M., Ferrara, M., Maltoni, D.: From gaming to research: GTA V for synthetic data generation for robotics and navigations (2025). arXiv preprint arXiv:2502.12303
17. Singh, R., Liu, J., Van Wyk, K., Chao, Y.-W., Lafleche, J.-F., Shkurti, F., Ratliff, N., Handa, A.: Synthetica: large scale synthetic data for robot perception (2024). arXiv preprint arXiv:2410.21153
18. Balloch, J.C., Agrawal, V., Essa, I., Chernova, S.: Unbiasing semantic segmentation for robot perception using synthetic data feature transfer (2018). arXiv preprint arXiv:1809.03676
19. Małek, K., Dybała, J., Kordecki, A., Hondra, P., Kijania, K.: Offroadsynth open dataset for semantic segmentation using synthetic-data-based weight initialization for autonomous UGV in off-road environments. J. Intell. Robot. Syst. **110**(2), 76 (2024)
20. Hart, K.M., Goodman, A.B., O'Shea, R.P.: Automatic generation of machine learning synthetic data using ROS. In: International

Conference on Human–Computer Interaction, pp. 310–325. Springer (2021)
21. Li, Z., Li, F., Zhang, W., Zheng, Z., Liu, X., Liu, Y., Zeng, L.: Thud++: large-scale dynamic indoor scene dataset and benchmark for mobile robots (2024). arXiv preprint arXiv:2412.08096
22. Garcia-Garcia, A., Martinez-Gonzalez, P., Oprea, S., Castro-Vargas, J.A., Orts-Escolano, S., Garcia-Rodriguez, J., Jover-Alvarez, A.: The robotrix: An extremely photorealistic and very-large-scale indoor dataset of sequences with robot trajectories and interactions. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6790–6797. IEEE (2018)
23. Camargo, C., Gonçalves, J., Conde, M.Á., Rodríguez-Sedano, F.J., Costa, P., García-Peñalvo, F.J.: Systematic literature review of realistic simulators applied in educational robotics context. Sensors **21**(12), 4031 (2021)
24. Vie, J.-J., Rigaux, T., Minn, S.: Privacy-preserving synthetic educational data generation. In: European Conference on Technology Enhanced Learning, pp. 393–406. Springer (2022)
25. Gonçalves, J., Lima, J., Oliveira, H., Costa, P.: Sensor and actuator modeling of a realistic wheeled mobile robot simulator. In: 2008 IEEE International Conference on Emerging Technologies and Factory Automation, pp. 980–985. IEEE (2008)
26. Coelho, J., Brancalião, L., Alvarez, M., Costa, P., Gonçalves, J.: Prototyping and control of an educational manipulator robot. In: 2024 10th International Conference on Control, Decision and Information Technologies (CoDIT), pp. 1814–1819 (2024). https://doi.org/10.1109/CoDIT62066.2024.10708583
27. Brancalião, L., Alvarez, M., Conde, M.Á., Costa, P., Gonçalves, J.: Towards a more accurate time of flight distance sensor to be applied in a mobile robotics application. In: International Conference on Technological Ecosystems for Enhancing Multiculturality, pp. 1145–1155. Springer (2022)
28. Brancalião, L., Alvarez, M., Coelho, J., Conde, M., Costa, P., Gonçalves, J.: Programming mobile robots in an educational context: a hardware-in-the-loop approach. In: 2024 10th International Conference on Control, Decision and Information Technologies (CoDIT), pp. 1820–1824 (2024). https://doi.org/10.1109/CoDIT62066.2024.10708336
29. Isermann, R., Schaffnit, J., Sinsel, S.: Hardware-in-the-loop simulation for the design and testing of engine-control systems. Control. Eng. Pract. **7**(5), 643–653 (1999)
30. Mihalič, F., Truntič, M., Hren, A.: Hardware-in-the-loop simulations: a historical overview of engineering challenges. Electronics **11**(15), 2462 (2022)
31. Woodman, S.J., Shah, D.S., Landesberg, M., Agrawala, A., Kramer-Bottiglio, R.: Stretchable Arduinos embedded in soft robots. Sci. Robot. **9**(94), 6844 (2024)
32. Electronics, P.R..: QTR-8A Reflectance Sensor Array (2024). https://www.pololu.com/product/960. Accessed 01 March 2024
33. Bartels, R.H., Beatty, J.C., Barsky, B.A.: An Introduction to Splines for Use in Computer Graphics and Geometric Modeling. Morgan Kaufmann, Burlington (1995)
34. Chen, L., Hontoir, Y., Huang, D., Zhang, J., Morris, A.J.: Combining first principles with black-box techniques for reaction systems. Control. Eng. Pract. **12**(7), 819–826 (2004). https://doi.org/10.1016/j.conengprac.2003.09.006
35. VL53L0X Datasheet—STMicroelectronics. https://www.alldatasheet.com/datasheet-pdf/pdf/948120/STMICROELECTRONICS/VL53L0X.html?mo. Accessed 08 Nov 2022
36. Gonçalves, J., Pinto, V.H., Costa, P.: A line follower educational mobile robot performance robustness increase using a competition as benchmark. In: 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), pp. 934–939 (2019). https://doi.org/10.1109/CoDIT.2019.8820556
37. Horio, M., Feng, Y., Kokado, T., Takasawa, T., Yasutomi, K., Kawahito, S., Komuro, T., Nagahara, H., Kagawa, K.: Resolving multi-path interference in compressive time-of-flight depth imaging with a multi-tap macro-pixel computational CMOS image sensor. Sensors **22**(7), 2442 (2022)
38. Brancalião, L., Conde, M.Á., Costa, P., Gonçalves, J.: Stochastic modeling of a time of flight sensor to be applied in a mobile robotics application. In: APCA International Conference on Automatic Control and Soft Computing, pp. 621–632. Springer (2022)
39. Kassow Robots—KR810. https://www.kassowrobots.com/products/kr810/. Accessed 08 Nov 2022
40. Kassow Robots: Software Manual. https://www.rd-as.com/app/uploads/2019/02/Software-manual-Kassow_Robots_V1.0.0_UPDATED-FONTS.pdf. Accessed 08 Nov 2022
41. Brancalião, L., Alvarez, M., Conde, M., Costa, P., Gonçalves, J.: Simulation model of a time of flight distance sensor using SimTwo. In: International Conference on Technological Ecosystems for Enhancing Multiculturality, pp. 572–581. Springer (2023)
42. Costa, P.: SimTwo—A Realistic Simulator for Robotics. https://github.com/P33a/SimTwo/
43. Smith, R.: Open Dynamics Engine (2007). https://www.ode.org/
44. Brancalião, L., Alvarez, M., Coelho, J., Conde, M., Costa, P., Gonçalves, J.: Learning mobile robotics: an approach based on a classroom competition. In: International Conference on Technological Ecosystems for Enhancing Multiculturality, pp. 648–658. Springer (2024)
45. Costa, P., Gonçalves, J., Lima, J., Malheiros, P.: SimTwo realistic simulator: a tool for the development and validation of robot software. Theory Appl. Math. Comput. Sci. **1**(1), 17–33 (2011)
46. Lima, J., Kalbermatter, R.B., Braun, J., Brito, T., Berger, G., Costa, P.: A realistic simulation environment as a teaching aid in educational robotics. In: 2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE), pp. 430–435. IEEE (2022)
47. Braun, J., Júnior, A.O., Berger, G.S., Lima, J., Pereira, A.I., Costa, P.: Robotatfactory 4.0: a ROS framework for the SimTwo simulator. In: 2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 205–210. IEEE (2022)
48. Gonçalves, J., Lima, J., Costa, P.J., Moreira, A.P.: Modeling and simulation of the emg30 geared motor with encoder resorting to SimTwo: the official robot@ factory simulator. In: Advances in Sustainable and Competitive Manufacturing Systems: 23rd International Conference on Flexible Automation and Intelligent Manufacturing, pp. 307–314. Springer (2013)
49. Lima, J., Costa, P., Brito, T., Piardi, L.: Hardware-in-the-loop simulation approach for the robot at factory lite competition proposal. In: 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 1–6. IEEE (2019)
50. Piardi, L., Eckert, L., Lima, J., Costat, P., Valente, A., Nakano, A.: 3D simulator with hardware-in-the-loop capability for the micromouse competition. In: 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 1–6. IEEE (2019)
51. Bacic, M.: On hardware-in-the-loop simulation. In: Proceedings of the 44th IEEE Conference on Decision and Control, pp. 3194–3198. IEEE (2005)
52. Suganuma, L.: Laiany/SyMoLo-dataset: SyMoLo_dataset_v1.0. https://doi.org/10.5281/zenodo.15806257