Electronic Research Archive

ERA, 31(4): 1789–1803.
DOI: 10.3934/era.2023092
Received: 04 November 2022
Revised: 13 January 2023
Accepted: 28 January 2023
Published: 07 February 2023

http://www.aimspress.com/journal/ERA

Research article

Differential drive kinematics and odometry for a mobile robot using TwinCAT

Miguel Ferreira[1], Luís Moreira[2] and António Lopes[1,2,*]

[1] Departamento de Engenharia Mecânica, Faculdade de Engenharia (FEUP), Universidade do Porto, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
[2] Instituto de Ciência e Inovação em Engenharia Mecânica e Engenharia Industrial (INEGI), Rua Dr. Roberto Frias, 4200-465 Porto, Portugal

* **Correspondence:** Email: aml@fe.up.pt.

Abstract: In this paper, we propose a motion control system for a low-cost differential drive mobile robot. The robotic platform is equipped with two driven wheels powered by Beckhoff motors, instrumented with incremental encoders. The control system is designed and implemented using Beckhoff's TwinCAT 3 automation software, running on an industrial PC. The system is tested and experimentally tuned to achieve optimal performance. The method allows addressing both odometry motion accuracy and motion correction in order to obtain minimum trajectory errors. Test results on linear and angular robot trajectories show errors below 0.02 and 0.03%, respectively, after tuning of the motion parameters. The proposed approach can be expanded, tweaked and applied to other differential drive TwinCAT 3 based robotic solutions. This will contribute to expanding mobile robot applications to a variety of fields, such as industrial automation, logistics, warehouse management, health care, ocean and space exploration and a variety of other industrial and non-industrial activities.

Keywords: mobile robotics; differential drive; kinematics; odometry; calibration; TwinCAT

1. Introduction

Robotics is a critical component of industrial development and automation. It enables significant growth in manufacturing efficiency, productivity, precision and quality, while also increasing safety when performing tasks that would be dangerous for a human operator to perform.

Automation solutions incorporating mobile robots in industry have grown in popularity in the last decades. This is due to an increasing demand for flexible solutions. Autonomous mobile robots are used in multiple areas, such as industrial automation, logistics and warehouse management, health care and services and many other industrial and non-industrial applications [1,2].

The basics of mobile robotics can be divided into locomotion, perception, localization and planning and navigation. Locomotion is required for the robot to move in its working environment. It relies on suitable motion control schemes that generate the necessary control actions to the motors' drivers, based on set points and feedback data [1–4].

There are many works focused on the analysis of kinematic models of differential drive vehicles [5–9]. Additionally, multiple studies show methods to test the accuracy of these models and how to modify them to reduce errors in odometry [10–13]. On the other hand, robot path planning and optimization either in static or dynamic environments represent key issues. Path planning strategies are often classified as (i) analytical, (ii) enumerative, (iii) meta-heuristic and (iv) evolutionary [14]. In particular, the evolutionary methods can be regarded as optimization procedures, and many algorithms have been proposed [15–22].

A quality control system is necessary to ensure that the robot can move with high accuracy, which is very important for industrial environments. Motion control systems for mobile robots with various drive types utilizing Beckhoff's TwinCAT automation software have been addressed in a small number of works. For instance, in [23] a new remote operated vehicle (ROV) control system based on TwinCAT was developed. The solution allowed the connection of functional modules and hardware and/or software modifications, while reducing the time needed for maintenance. A graphical user interface (GUI) was also created for real-time data monitoring and controlling the ROV. In [24] a heavy-duty omni-directional Mecanum robot was presented, together with its control system and simulation design. The control system combined a Beckhoff module and the Robot Operating System (ROS) for low-level motion execution and high-level intelligent navigation tasks, respectively. In addition, a virtual simulation environment was validated. In [25] real-time sensor fusion for smooth position feedback of a heavy-duty field robot was addressed. The robot autonomous driving system was based on a Beckhoff hardware platform running TwinCAT. The programming was performed in a MATLAB/Simulink environment, while the functional modules were built on a target TwinCAT system. The robot used wireless communication between the real-time hardware and the host PC. In [26] a solution for a network of real-time-capable modules was derived and illustrated on a prototype of a mobile robot using TwinCAT CNC. In [27] a novel wheel-legged robotic system was developed. The robot control software was implemented using TwinCAT. In [28] a path planning and navigation control system was developed for a 12 m length driverless electric bus. The TwinCAT software system was utilized for path planning and trajectory tracking controller, while ensuring real-time update and synchronization rate. However, to the best of the authors' knowledge, none of these TwinCAT implementations has addressed motion accuracy testing and odometry of the robot, nor motion correction in order to obtain minimum errors. On the other hand, the proposed approach is flexible, completely modular and extendable to accommodate different sensors and actuators, as well as other robotic differential drive platforms utilizing TwinCAT 3.

The present paper proposes a motion control scheme for a low-cost differential drive mobile robot. The robot locomotion platform consists of a differential drive setup using two wheels powered by Beckhoff motors and a free caster wheel. The motion control is developed using Beckhoff's TwinCAT 3 automation software in an industrial PC (IPC). The system is tested and experimentally tuned to

achieve excellent performance. Field tests of the robot performing linear and angular trajectories reveal errors below 0.02 and 0.03%, respectively, after tuning its motion parameters. The proposed approach can be extended and applied to other robotic differential drive platforms utilizing TwinCAT 3, being a cost-effective solution for real-world applications.

The paper is organized as follows: Section 2 describes the robot and its hardware; Section 3 presents the development of the motion control system, the kinematics of the robot and the testing and correction of the odometry. Finally, Section 4 outlines the main conclusions.

2. Robot overview

The mobile robot uses a differential drive platform with two independently driven wheels and one free caster wheel. Each of the driven wheels is equipped with an AS1050-0120 Beckhoff motor with an incorporated incremental encoder. The resolution of the encoders is 1.8° or 200 increments per revolution. The motors are coupled with a gearbox with a gear ratio of 5. The robotic platform is also equipped with an anthropomorphic robot arm, which is not addressed in this paper. A picture of the robot and the two independently driven wheels can be seen in Figure 1, along with their respective Beckhoff stepper motors and gearboxes [29].

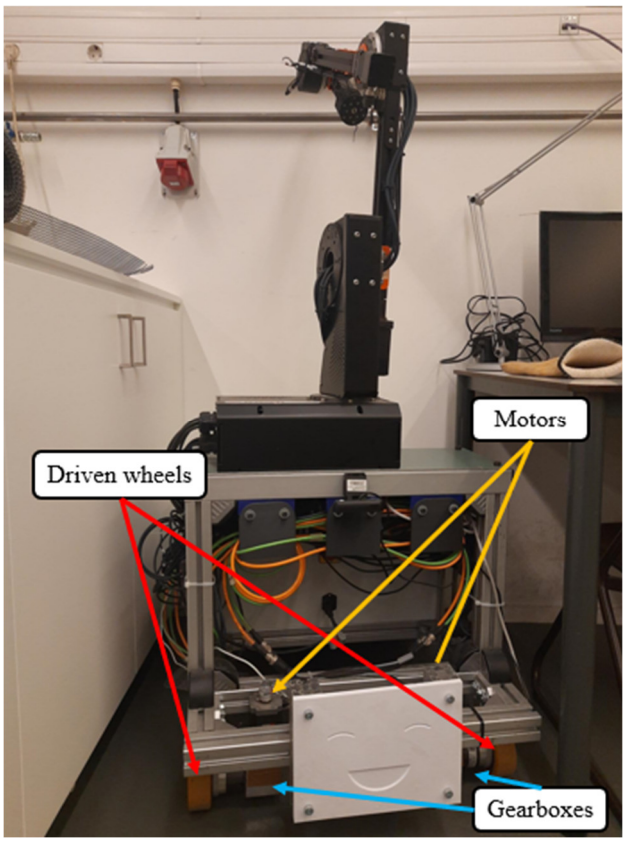

Figure 1. Front picture of the robot.

The electrical cabinet can be seen in Figure 2. It contains three different power supplies, one with 48 V, one with 24 V and one with 5 V:

1) The 48 V power supply is used to power 5 EtherCAT terminals (3 EL7047 and 2 EL7041) that connect to 3 stepper motors used in the robot arm and 2 stepper motors used for the robot's locomotion.
2) The 24 V power supply is used to power the EtherCAT terminal EL9800, which supplies 2 EtherCAT EL7037 modules that connect to 2 other stepper motors used in the robot arm. It also powers their encoders, the robot's IPC, the EtherCAT coupler EK1100 and the input (EL1008) and output terminals (EL2008).
3) The 5 V power supply is used for three of the encoders in the robot arm and for the two encoders used for locomotion.

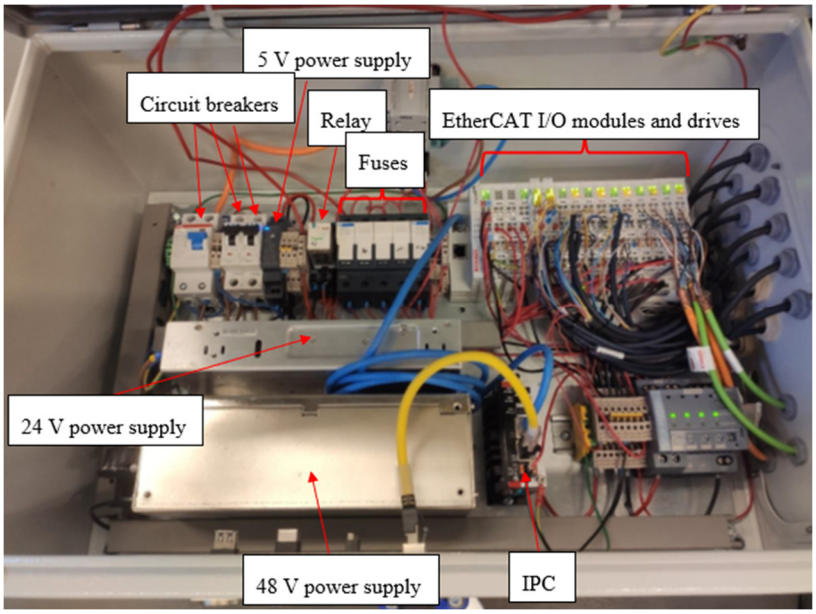

Figure 2. Electrical cabinet of the robot.

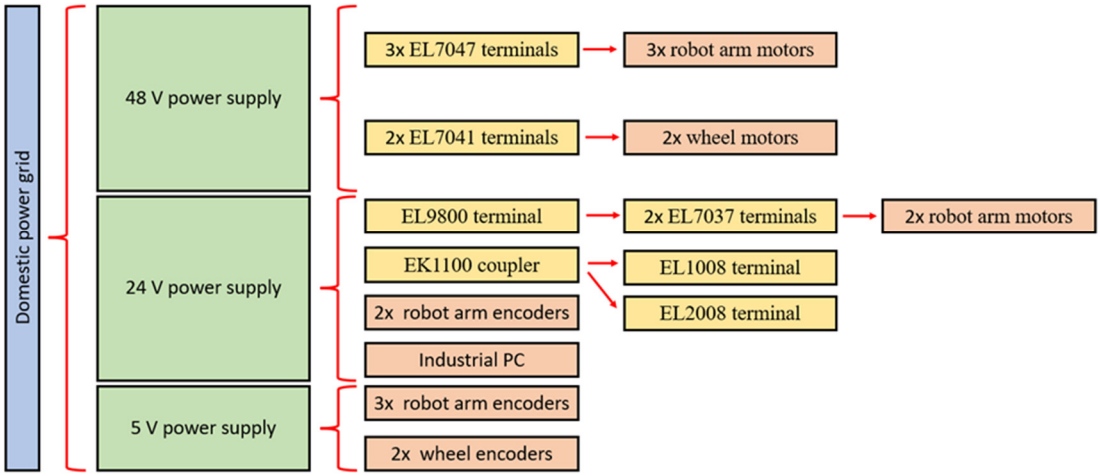

Figure 3. Electrical power schema of the robot.

Figure 3 presents a schema of the electrical power used in the robot. The IPC used in this robot is the C6015-0010 model from Beckhoff, running with Windows 10 as its operating system and with TwinCAT 3 installed. The robot is also equipped with an emergency button that, when activated, turns

off all circuits that send power to the motors, while keeping the command circuits on, including the IPC and the EtherCAT communication.

3. Motion control system

The software used for the control and command of the robotic system is TwinCAT 3. TwinCAT 3 is an automation software solution for PC-based control that can turn a PC with EtherCAT communication into a real time controller with multiple subsystems running simultaneously. It consists of a XAR (eXtended Automation Runtime) that runs the already programmed control modules and the XAE (eXtended Automation Engineering) that, using Microsoft Visual Studio, allows the user to setup, modify and program the control system to be used.

TwinCAT 3 uses a number of modules that communicate with one another to form a complete controller. There are modules for the system configuration (*SYSTEM*), motion control (*MOTION*), PLC programming (*PLC*), safety (*SAFETY*), C++ programming (*C++*), analytics (*ANALYTICS*) and I/O configuration (*I/O*). This paper does not make use of the SAFETY, C++ or ANALYTICS modules.

When configuring the system, the software in the I/O module recognized all connected EtherCAT devices. Furthermore, the motor drivers were configured, which included entering multiple motor properties, such as maximum current, nominal voltage, number of steps per revolution and encoder resolution. The different movement axes were configured in the MOTION module. This module is responsible for the numerical control of the movement axes. It can use point-to-point control to generate the signals needed to send to the appropriate I/O device. It has several tabs where different fields must be configured correctly. The PLC module implements the system logic and is responsible for the overall control of the robot's motion system. This module is programmable according to the IEC 61131-3 international standard for programmable PLCs, which supports the following programming languages: ladder diagram (LD), function block diagram (FBD), sequential function chart (SFC), structured text (ST) and instruction list (IL). In the case of this paper, every program in the *PLC* module was developed in ST.

The robot used in this paper was also simultaneously used for the development of an autonomous navigation system using ROS. As such, the objective was for TwinCAT to receive velocity commands from the ROS system while simultaneously sending odometry information to the ROS environment. This information flow is shown in Figure 4.

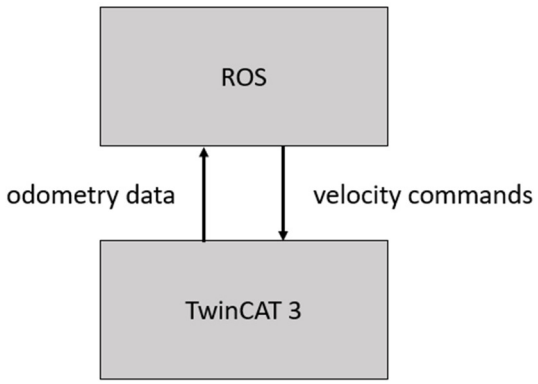

Figure 4. Flow of information between TwinCAT 3 and ROS.

The ROS Navigation Stack uses the *move_base* package. This package generates the following commands: linear velocity in x (V_x), linear velocity in y (V_y) and angular velocity in θ (V_θ). These directions are relative to the robot's current position, in accordance with Figure 5. This specific robot can only have velocities in the x and θ directions since it is a differential drive robot. Because there are no other possible velocities that a differential drive robot can take, V_x and V_θ are also respectively known as linear and angular velocity.

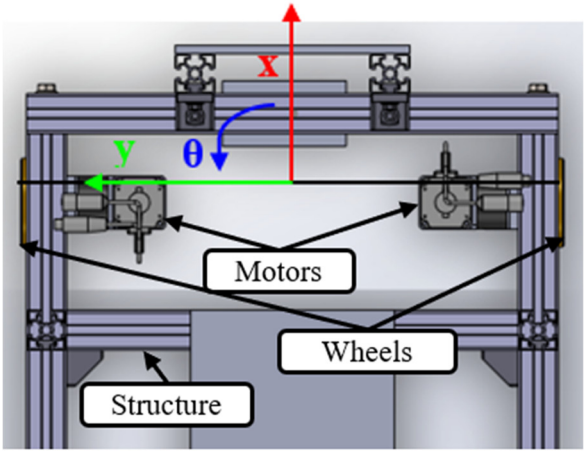

Figure 5. Robot's movement axes.

3.1. Differential drive kinematics

Because TwinCAT 3 controls the movement of both wheel axes independently, the robot's linear and angular velocities must be converted into velocities for the left and right wheels. Furthermore, odometry information about the robot must be obtained using information from the encoders.

TwinCAT's *MOTION* module was configured with information about the motors and transmission systems used. As a result, it can use wheel velocities specified in mm/s and automatically read information from encoders in mm. As such, it is only necessary to transform V_x and V_θ into V_R (velocity of right wheel) and V_L (velocity of left wheel), and δ_R and δ_L into δ_x, δ_y and δ_θ, where δ means the variation of position between each control cycle.

In differential drives, linear and angular velocities are given as functions of the velocities of the wheels, according to [5]:

$$V_x = \frac{V_R + V_L}{2} \tag{1}$$

$$V_\theta = \frac{V_R - V_L}{B} \tag{2}$$

where B is the axial distance between the driven wheels. Adding and rearranging Eqs (1) and (2) results in

$$V_R = V_x + \frac{V_\theta \cdot B}{2} \tag{3}$$

$$V_L = (2 \cdot V_x) - V_R \tag{4}$$

Since the increments in position of each wheel can be obtained in mm from the encoder data in TwinCAT, linear and angular displacements of the robot are given by

$$\delta_x = \frac{\delta_R + \delta_L}{2} \tag{5}$$

$$\delta_\theta = \frac{\delta_R - \delta_L}{B} \tag{6}$$

The length of the red arc in Figure 6 is given by δ_x (the size is exaggerated for purposes of easier understanding). The linear distance, d, between the two points (green line) can be obtained with the trigonometric relations:

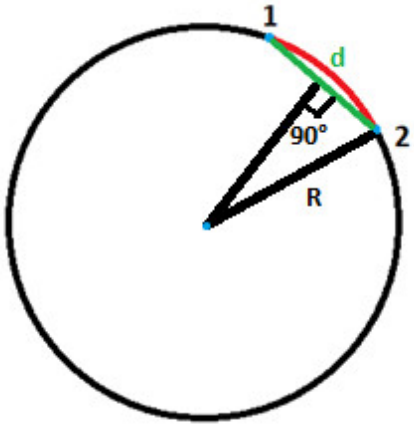

Figure 6. Arc traveled by the robot between two points.

$$d = 2 \cdot R \cdot \sin\left(\frac{\delta_\theta}{2}\right) \tag{7}$$

$$R = \frac{\delta_x}{\delta_\theta} \tag{8}$$

By adding the previous two equations, Eq (9) is obtained:

$$d = 2 \cdot \frac{\delta_x}{\delta_\theta} \cdot \sin\left(\frac{\delta_\theta}{2}\right) \tag{9}$$

This equation, however, can only be used if the robot did not move in a straight line between two cycles of the program. If δ_θ is 0, then Eq (10) is used instead:

$$d = \delta_x \tag{10}$$

The position of the robot in relation to its starting frame can be obtained over time by adding the linear and angular displacements between two cycles of the program ($k + 1$ and k) to the last calculated position of the robot, using

$$\theta_{k+1} = \theta_k + \delta_\theta \tag{11}$$

$$x_{k+1} = x_k + \delta_x \cdot \cos\left(\frac{\theta_k + \theta_{k+1}}{2}\right) \tag{12}$$

$$y_{k+1} = y_k + \delta_x \cdot sin\left(\frac{\theta_k + \theta_{k+1}}{2}\right) \tag{13}$$

The angle used to calculate x and y is the average value of the angular position in each of the two cycles to reduce errors. Initial position is initialized to (0, 0, 0). Since usually these cycles are calculated at very high frequencies, the distance between two points in an arc given by Eq (9) can be approximated by the length of the arc itself with negligible error. Lower cycle frequency and higher linear and angular velocities result in a higher error when using this approximation. Herein, the cycle time of the program is 10 ms. At the robot's maximum linear velocity (V_x) of 0.3 m/s and maximum angular velocity (V_x) of 0.7 rad/s, the length of the arc traveled between each cycle (δ_x) is 3 mm, and the angular displacement (δ_θ) is 0.007 rad. Using these values in Eq (9) results in a linear distance traveled (d) of approximately 2.999994 mm. This means that, in this case, the maximum error obtainable from the use of this approximation is around 0.0002%.

3.2. Implementation in TwinCAT 3

The kinematics was implemented in a program developed in ST language in TwinCAT. It was necessary to take the velocity commands and make the wheels move at the respective wheel velocities until a new velocity command is obtained. To achieve this, the *MC_MoveVelocity* function block was used. This function block starts a continuous movement with specified velocity and direction. However, this function block cannot receive a null velocity as an input, which means that an additional function block must be used for when the velocity of any of the wheels is zero. As such, the *MC_Halt* function block was used for these cases, as it stops an axis's movement with a defined breaking ramp.

Since the *MC_MoveVelocity* function block only takes velocity inputs larger than 0 and a forward or negative direction value, IF conditions are written for all nine possible combinations of positive, null, and negative velocity values of each wheel. The velocity command variables V_x and V_θ and the odometry variables x, y and θ are all set as global variables to allow them to be written and read by ROS. The program also reads the current velocity of each wheel and calculates the linear and angular velocities the robot is moving at currently, as these are necessary odometry data for ROS. The odometry is calculated using the difference between the absolute values of the encoders in the current and last cycles. During the first cycle of the program, odometry values are set to (0, 0, 0), the motors are powered, and their axes are reset.

Because the *MC_MoveVelocity* and *MC_Halt* function blocks require a rising edge signal of the *Execute* Boolean input, it is also necessary to add the following lines present in Figure . These lines feed the mentioned function blocks an input of "Execute = FALSE". This allows the robot's movement to be flexible and able to change velocities at any moment.

```
61      //allow new movement commands to be executed
62      MoveR(Axis:=Axis_DIR, Execute:=FALSE);
63      MoveL(Axis:=Axis_ESQ, Execute:=FALSE);
64      StopR(Axis:=Axis_DIR, Execute:=FALSE);
65      StopL(Axis:=Axis_ESQ, Execute:=FALSE);
```

Figure 7. Necessary lines to allow new movement commands.

4. Odometry testing and correction

In order to minimize the trajectory errors, the values of the robot parameters need to be properly determined. This problem is challenging, but a deep analysis is out of the scope of this paper. As such, herein, an empirical tuning approach was adopted based on experimental testing and error analysis, by comparing the actual odometry trajectories with those recorded by an external measurement system. Naturally, this method is simple, but it does not guarantee optimal robot behavior. Different methods, namely, parameters' optimization based on evolutionary algorithms, can be an effective means to address the issue [30,31].

The quality, and magnitude of the errors, of the implemented odometry were tested by comparing the odometry values with the real trajectory of the robot. As such, a pen was attached to the robotic platform at the robot's rotation axis. A component to attach the pen to the robot in the position of its frame was modeled and 3D printed (Figure 8). Its geometry only allows the pen to move in the vertical axis, ensuring that it can stay in contact with the ground while the robot moves. A spring was inserted inside to guarantee that there is tension between the pen's ballpoint and the ground.

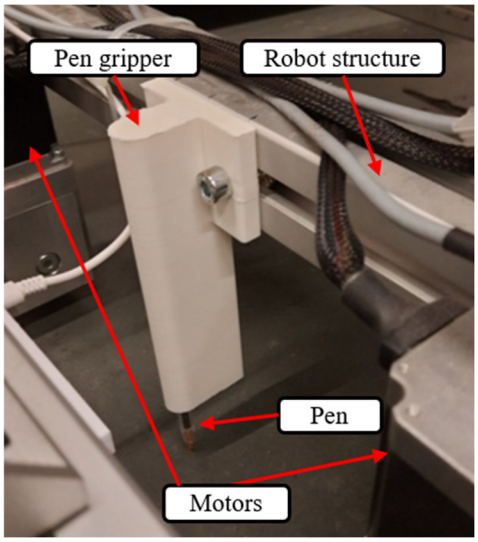

Figure 8. Pen gripper installed on the robot.

To make the robot perform specific movements that can be repeated, another TwinCAT program was also written in ST. This program includes the odometry functionalities already implemented, but the robot movements are specified for a set distance and relative to the position at the start of the

movement. To achieve this, the program utilizes the *MC_MoveRelative* function block. It can be used to perform linear, circular or rotational trajectories.

The quality of the linear odometry was tested by measuring the length of the line drawn when instructing the robot to move 1 m forward (Figure 9). This test was done at a linear velocity of 0.025 m/s and repeated three times. The results of the linear odometry tests are shown in Table 1.

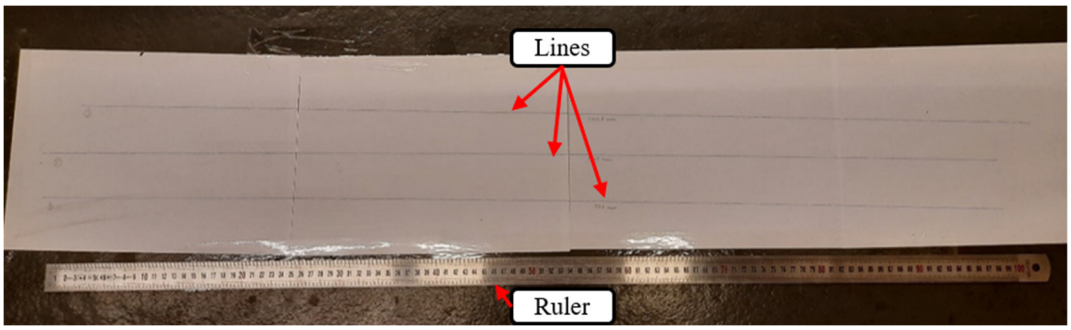

Figure 9. Picture of the lines produced by the robot and ruler used to measure them.

Table 1. Results of the initial linear odometry tests.

	x given by the odometry	Length of line	Error	%Error
Test 1	1000.0 mm	1009.5 mm	-9.5 mm	-0.94%
Test 2	1000.0 mm	1011.0 mm	-11.0 mm	-1.09%
Test 3	1000.0 mm	1010.5 mm	-10.5 mm	-1.04%
Average	1000.0 mm	1010.3 mm	-10.3 mm	-1.02%

The results of the linear odometry test show that the robot's odometry was giving values lower than the measured values by around 1.03%. To counteract this error, the scaling factor parameter of the motor's encoder was changed in TwinCAT. This scaling factor specifies how much each wheel moves in mm per increment of the encoder. The scaling factor is given by:

$$SF = \frac{\pi \cdot D}{4 \cdot e} \cdot i \tag{14}$$

where *SF* is the scaling factor, *D* is the diameter of the wheel, *i* is the gear ratio of the transmission system, and *e* is the number of encoder increments per rotation.

To correct the linear odometry of the robot, it is necessary to divide the scaling factor by (1 + %Error):

$$SF_{corrected} = \frac{SF}{1+\%Error} \tag{15}$$

Since *e* and *i* are constant, any changes to the SF can also be interpreted as changes to the wheel diameter considered. The original *SF* used in this robot was 0.015340 mm/Inc (mm per increment of the encoder). This was obtained using a wheel diameter of 100 mm in Eq (14). In this case, using Eq (15) a corrected *SF* of 0.015498 mm/Inc is obtained. This equates to considering a wheel diameter of around 101.0 mm. The previous linear odometry tests were repeated using the corrected scaling

factor. The results are shown in Table . We verify now an average error of around 0.02%, or 0.2 mm, which is within the resolution of the ruler used to measure the line length.

Table 2. Results of the linear odometry tests after correction.

	x given by the odometry	Length of line	Error	%Error
Test 1	1000.0 mm	1000.0 mm	0.0 mm	0.0%
Test 2	1000.0 mm	1000.0 mm	0.0 mm	0.0%
Test 3	1000.0 mm	1000.5 mm	-0.5 mm	-0.05%
Average	1000.0 mm	1000.2 mm	-0.2 mm	-0.02%

Angular odometry tests were performed after the correction of the linear odometry with the new scaling factor. The robot was programmed to perform an in-place rotation of 360°. A line parallel to the wheel's initial and final positions was drawn on a piece of paper on the floor. The angle between these two lines was measured and compared to the angle given by the robot's odometry. This test was done at an angular velocity of 0.182 rad/s and repeated three times. The results are summarized in Table.

Table 3. Results of the first series of angular odometry tests.

	θ given by the odometry	Angle measured	Error	%Error
Test 1	360.0°	363.1°	-3.1°	-0.85%
Test 2	360.0°	362.9°	-2.9°	-0.80%
Test 3	360.0°	361.8°	-1.8°	-0.50%
Average	360.0°	362.6°	-2.6°	-0.72%

Angular displacement is given by Eq (6). Considering that the linear odometry is already corrected, angular odometry can only be corrected by changing the value of the distance between wheels (B), which is inversely proportional to the angular displacement. It can be corrected by:

$$B_{corrected} = B \cdot (1 + \%Error) \tag{16}$$

In this case, the measuring error was considered too large in comparison with the odometry error to take any conclusions. Because of this, a new series of tests was conducted by commanding the robot to perform a 1080° in-place rotation with the same angular velocity of 0.182 rad/s. This larger movement results in an increased odometry error, while maintaining similar measurement errors, which allows for a more accurate percentual error to be calculated. Results of these tests are shown in Table 4.

By applying the average error obtained from these tests to Eq (16) using the original value of B of 550 mm, a corrected distance between wheels of 545.38 mm is determined. The same angular odometry tests were repeated using the corrected value of B. The results are shown in Table 5.

Table 4. Results of the second series of angular odometry tests.

	θ given by the odometry	Angle measured	Error	%Error
Test 1	1080.0°	1089.1°	-9.1°	-0.84%
Test 2	1080.0°	1090.0°	-10.0°	-0.92%
Test 3	1080.0°	1088.5°	-8.5°	-0.78%
Average	1080.0°	1089.2°	-9.2°	-0.84%

Table 5. Results of the angular odometry tests after correction.

	θ given by the odometry	Angle measured	Error	%Error
Test 1	1080.0°	1079.2°	+0.8°	+0.07%
Test 2	1080.0°	1079.8°	+0.2°	+0.02%
Test 3	1080.0°	1080.1°	-0.1°	-0.01%
Average	1080.0°	1079.7°	+0.3°	+0.03%

In a 1080° trajectory, the results now indicate an average error of 0.03°, or approximately 0.03%, which is within the repeatability errors of the tests. As a result, the value of 545.38 mm for the corrected distance between wheels is accepted.

When using different values for the velocities, the odometry errors appeared to be similar due to low wheel slippage occurring in the testing environment.

5. Conclusions

A motion control system for a low-cost differential drive mobile robot was proposed, based on Beckhoff's TwinCAT 3 automation software running on an IPC. An empirical tuning approach based on experimental testing and error analysis, by comparing the odometry trajectories with those measured with an external measurement system, was implemented. The approach led to good robot performance, with errors below 0.02 and 0.03% while performing linear and angular trajectories, respectively. The method is simple, but it does not guarantee optimal robot behavior. Further work will address parameter optimization, namely, the adoption of evolutionary algorithms, and additional tests to further assess the accuracy of the system, including verifying the accuracy of each wheel individually and testing differences between clockwise and counterclockwise movement. The proposed approach is flexible and modular, and it can be extended, adjusted and used in other robotic differential drive based solutions utilizing TwinCAT 3.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. R. Siegwart, I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, Cambridge, Massachusetts: The MIT Press, 2004.
2. U. Nehmzow, *Mobile robotics: A Practical Introduction*, 2nd edition, Springer, (2012), 1–21.
3. F. Rubio, F. Valero, C. Llopis-Albert, A review of mobile robots: concepts, methods, theoretical framework, and applications, *Int. J. Adv. Rob. Syst.*, **16** (2019). https://doi.org/10.1177/1729881419839
4. L. Jaulin, *Mobile Robotics*, 2nd edition, London: ISTE Ltd; John Wiley & Sons, 2019. https://doi.org/10.1002/9781119663546
5. S. K. Malu, J. Majumdar, Kinematics, localization and control of differential drive mobile robot, *Global J. Res. Eng.*, **14** (2014), 1–7. Available from: https://engineeringresearch.org/index.php/GJRE/article/view/1233.
6. G. Díaz-García, L. F. Giraldo, S. Jimenez-Leudo, Dynamics of a differential wheeled robot: control and trajectory error bound, in *2021 IEEE 5th Colombian Conference on Automatic Control (CCAC)*, (2021), 25–30. https://doi.org/10.1109/CCAC51819.2021.9633318
7. T. Hellström, *Kinematics Equations for Differential Drive and Articulated Steering*, Department of Computing Science, Umeå University, 2011. Available from: https://www.diva-portal.org/smash/get/diva2:796235/FULLTEXT01.pdf.
8. E. Maulana, M. A. Muslim, A. Zainuri, Inverse kinematics of a two-wheeled differential drive an autonomous mobile robot, in *2014 Electrical Power, Electronics, Communicatons, Control and Informatics Seminar (EECCIS)*, (2014), 93–98. https://doi.org/10.1109/EECCIS.2014.7003726
9. F. A. Salem, Dynamic and kinematic models and control for differential drive mobile robots, *Int. J. Curr. Eng. Technol.*, **3** (2013), 253–263. Available from: https://inpressco.com/wp-content/uploads/2013/03/Paper6253-2632.pdf.
10. J. Borenstein, H. R. Everett, L. Feng, Where am I? Sensors and methods for mobile robot positioning, The University of Michigan, 1996. Available from: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=80913ec26b835fd8238fe95c90c67770d528f9f9.
11. K. S. Chong, L. Kleeman, Accurate odometry and error modelling for a mobile robot, in *Proceedings of International Conference on Robotics and Automation*, **4** (1997), 2783–2788. https://doi.org/10.1109/ROBOT.1997.606708
12. J. Borenstein, L. Feng, Measurement and correction of systematic odometry errors in mobile robots, *IEEE Trans. Rob. Autom.*, **12** (1996), 869–880. https://doi.org/10.1109/70.544770
13. R. B. Sousa, M. R. Petry, A. P. Moreira, Evolution of odometry calibration methods for ground mobile robots, in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, (2020), 294–299. https://doi.org/10.1109/ICARSC49921.2020.9096154
14. M. N. Zafar, J. C. Mohanta, Methodology for path planning and optimization of mobile robots: a review, *Procedia Comput. Sci.*, **133** (2018), 141–152. https://doi.org/10.1016/j.procs.2018.07.018
15. B. K. Patle, G. Babu L, A. Pandey, D. Parhi, A. Jagadeesh, A review: on path planning strategies for navigation of mobile robot, *Def. Technol.*, **15** (2019), 582–606. https://doi.org/10.1016/j.dt.2019.04.011

16. Q. B. Zhang, P. Wang, Z. H. Chen, An improved particle filter for mobile robot localization based on particle swarm optimization, *Expert Syst. Appl.*, **135** (2019), 181–193. https://doi.org/10.1016/j.eswa.2019.06.006
17. W. Deng, J. Xu, X. Z. Gao, H. Zhao, An enhanced MSIQDE algorithm with novel multiple strategies for global optimization problems, *IEEE Trans. Syst. Man Cybern.: Syst.*, **52** (2022), 1578–1587. https://doi.org/10.1109/TSMC.2020.3030792
18. Y. Song, X. Cai, X. Zhou, B. Zhang, H. Chen, Y. Li, et al., Dynamic hybrid mechanism-based differential evolution algorithm and its application, *Expert Syst. Appl.*, **213** (2023), 118834. https://doi.org/10.1016/j.eswa.2022.118834
19. W. Deng, J. Xu, H. Zhao, Y. Song, A novel gate resource allocation method using improved PSO-based QEA, *IEEE Trans. Intell. Transp. Syst.*, **23** (2022), 1737–1745. https://doi.org/10.1109/TITS.2020.3025796
20. W. Deng, X. Zhang, Y. Zhou, Y. Liu, X. Zhou, H. Chen, et al, An enhanced fast non-dominated solution sorting genetic algorithm for multi-objective problems, *Inf. Sci.*, **585** (2022), 441–453. https://doi.org/10.1016/j.ins.2021.11.052
21. H. Chen, F. Miao, Y. Chen, Y. Xiong, T. Chen, A hyperspectral image classification method using multifeature vectors and optimized KELM, *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, **14** (2021), 2781–2795. https://doi.org/10.1109/JSTARS.2021.3059451
22. Y. Yu, Z. Hao, G. Li, Y. Liu, R. Yang, H. Liu, Optimal search mapping among sensors in heterogeneous smart homes, *Math. Biosci. Eng.*, **20** (2023), 1960–1980. https://doi.org/10.3934/mbe.2023090
23. M. Aliff, N. Raihan, I. Yusof, N. Samsiah, Development of remote operated vehicle (ROV) control system using Twincat at main control pod (MCP), *Int. J. Innovative Technol. Exploring Eng.*, **8** (2019), 12. https://doi.org/10.35940/ijitee.L4019.1081219
24. L. Xie, C. Scheifele, W. Xu, K. A. Stol, Heavy-duty omni-directional Mecanum-wheeled robot for autonomous navigation: system development and simulation realization, in *2015 IEEE International Conference on Mechatronics (ICM)*, (2015), 256–261. https://doi.org/10.1109/ICMECH.2015.7083984
25. H. Liikanen, M. M. Aref, J. Mattila, M-Estimator application in real-time sensor fusion for smooth position feedback of heavy-duty field robots, in *2019 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, (2019), 368–373. https://doi.org/10.1109/CIS-RAM47153.2019.9095821
26. C. Scheifele, A. Lechler, C. Daniel, W. Xu, Real-time extension of ROS based on a network of modular blocks for highly precise motion generation, in *2016 IEEE 14th International Workshop on Advanced Motion Control (AMC)*, (2016), 129–134. https://doi.org/10.1109/AMC.2016.7496339
27. J. He, Y. Sun, L. Yang, J. Sun, Y. Xing, F. Gao, Design and control of TAWL—A wheel-legged rover with Terrain-adaptive wheel speed allocation capability, *IEEE/ASME Trans. Mechatron.*, **27** (2022), 2212–2223. https://doi.org/10.1109/TMECH.2022.3176638
28. L. Yu, D. Kong, X. Shao, X. Yan, A path planning and navigation control system design for driverless electric bus, *IEEE Access*, **6** (2018), 53960–53975. https://doi.org/10.1109/ACCESS.2018.2868339
29. T. da R. Anjo, Implementação e controlo de um robô móvel com braço antropomórfico, 2021. Available from: https://repositorio-aberto.up.pt/bitstream/10216/136886/2/507140.pdf.

30. W. Deng, H. Liu, J. Xu, H. Zhao, Y. Song, An improved quantum-inspired differential evolution algorithm for deep belief network, *IEEE Trans. Instrum. Meas.*, **69** (2020), 7319–7327. https://doi.org/10.1109/TIM.2020.2983233
31. W. Deng, L. Zhang, X. Zhou, Y. Zhou, Y. Sun, W. Zhu, et al., Multi-strategy particle swarm and ant colony hybrid optimization for airport taxiway planning problem, *Inf. Sci.*, **612** (2022), 576–593. https://doi.org/10.1016/j.ins.2022.08.115

AIMS Press