ISSN 2007-9737

Position and Orientation Control on Three-step by a State Machine for an Omnidirectional Mobile Robot

Prometeo Cortés-Antonio*, Fevrier Valdez, Patricia Melin, Oscar Castillo

Tecnológico Nacional de México/IT Tijuana,
Mexico

prometeo.cortes@tectijuana.edu.mx, {fevrier, pmelin, ocastillo}@tectijuana.mx

Abstract. In the position and orientation control (pose) of a mobile robot, a trajectory is created, and the robot starts from its initial position and ends in a desired position and orientation. Most state-of-the-art solutions employ control laws based on classic Proportional-Integral-Derivative controllers and/or variants, fuzzy controllers, etc., which generate trajectories that can describe sharp curves or undesirable behaviors (for example, going backward). These curves are drastically dependent on whether the desired orientation is in front of the robot's initial pose. Therefore, a three-step control strategy for pose control of a mobile robot is presented. The first task is an orientation control which drives the robot to the desired position, followed by a position control (which can be performed over a single direction), and finally, another orientation control that drives the robot to the desired orientation.

Keywords: Control pose, position and orientation control, state machine, omnidirectional robot.

1 Introduction

Mobile autonomous robots are currently being applied to perform a wide variety of tasks, such as in the assistance of housework, offices, warehouses, the medical sector, etc. The Robotic Vacuum Cleaner Market was assessed at USD 4.5 billion in 2023 and it is anticipated to grow by 6.5 in 2024 according to Global Markin Insight.

On the other hand, it is already very common to hear the application of these, as assistants in distribution centers at companies such as Amazon, and other applications are found in the pharmaceutical industry, semiconductor plants, and food factories, just to mention a few [1-3].

Position control, and pose control (position and orientation) are some of the most common and studied tasks in the literature on autonomous mobile robots. A study of the pose control of an omnidirectional robot is carried out using different control techniques such as PI, PD, and Feed-Forward (FF) control to follow trajectories described by parametric polynomials [4].

A control strategy that uses computer vision techniques, known as serving, is presented in [5] o generates trajectories given an initial position and a desired position of an omnidirectional robot, and it uses a fuzzy system to change the values of the controllers' gains.

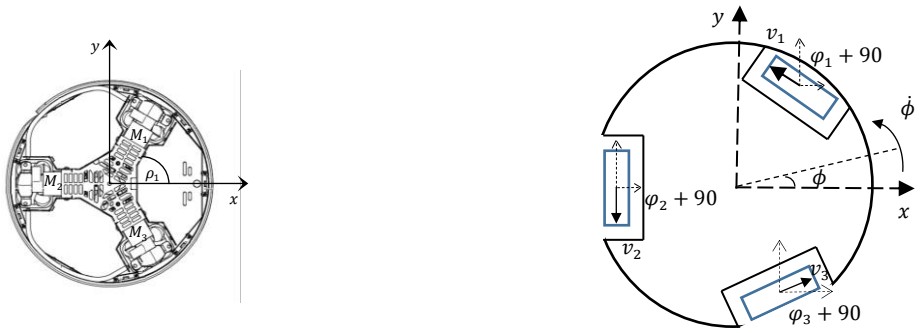
The application of pose control task using an omnidirectional robot as a movement assistant in the rehabilitation of arms and shoulders of cerebrovascular patients is shown [6], where the pose of the robot is estimated using fusion techniques of a Kalman filter, encoders, and images obtained from an external camera, and through a cascade position control are used to track reference points from generated trajectories.

On the other hand, there are works focused on different tasks, such as trajectory tracking [7], obstacle avoidance [8], optimization of control systems [9-10], and navigation applications [11-12].

Therefore, the creation of intelligent strategies that generate efficient movements of mobile robots is an emerging and interesting topic.

This article proposes a three-step position and orientation control strategy using a state machine, whose main behavior that differs from others is to generate translational movements of the robot that are always directed toward its front, to eliminate undesirable sharp curves.

This strategy is very convenient in cases where the robot has a front camera, and it also helps to simplify the laws of control, since the strategy



a) b)

Fig. 1. Geometric characteristics of Robotino omnidirectional robot. a) Position of the motors according to the body frame. B) Indicates the steering speed generated by the engines

divides the total movement into independent movements.

The structure of the article is as follows, in Section 1, an introduction to mobile autonomous robots and pose control applications for omnidirectional robots was presented. Section 2 will present the kinematics of a three-wheeled omnidirectional robot. Section 3 presents a general pose control strategy used for pose control, and section 4 presents the new pose control strategy proposed in this article, and extends the proposal into the autonomy problem using a heuristic-based autonomy. Section 5 shows an example of different pose control situations, and presents a couple of examples of pose control using the heuristic-based autonomy examples. Finally, Section 6 presents the conclusions and future work.

2 Omnidirectional Robot Kinematics

The main characteristic of omnidirectional (holonomic) robots is based on their ability to perform independent movements on the general maneuverability of mobile robots i.e., independent movements in each of its directions in a xy-plane, and its rotational movements. In general, omnidirectional robots can be configured with 3 or 4 wheels, and use unconventional wheels of the omnidirectional or mecanum type [13]. These types of wheels differ from conventional wheels by the rollers attached around their wheels. An example of an omnidirectional robot is the Robotino, manufactured by the Festo company, designed with a round chassis and three omnidirectional wheels.

The geometric characteristics of Robotino are shown in Figure 1, the location of the three motors that allow the wheel to be driven is observed [14]. The distance from the center of the robot chassis to each motor is R and the motors are separated from each other by an angular distance of 120 degrees.

According to the robot's frame, the motors are located in $\rho_1 = 60,\ \rho_2 = 180, \rho_3 = -60$, and the speeds generated by the motors $\upsilon_1, \upsilon_2, \upsilon_3$ have a rotated direction of 90-degree of the angle of the motors.

The kinematic model describing the wheel speeds of the omnidirectional robot is presented in equation (1):

$$\begin{bmatrix} \upsilon_1 \\ \upsilon_2 \\ \upsilon_3 \end{bmatrix} = \begin{bmatrix} -\sin(\phi + \rho_1) & \cos(\phi + \rho_1) & R \\ -\sin(\phi + \rho_2) & \cos(\phi + \rho_2) & R \\ -\sin(\phi + \rho_3) & \cos(\phi + \rho_3) & R \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix}. \quad (1)$$

Robotino is equipped with a large number of sensors, such as ultrasonic sensors, cameras, and optocouplers among others, which make it a good didactic tool for learning and research.

In addition, this robotic system has software tools for programming and simulating and can be programmed using different computational languages. [14-15].

3 Pose Control

The pose control problem can be defined as the task of moving a robot to a desired position and orientation. This task differs from position control since position control is not concerned with the final orientation of the robot.

A pose control strategy, based on Lyapunov's stability theorem for a unicycle robot, (bicycle, or differential drive with some adjustments), drives the robot asymptotically to the desired pose, and by using a polar coordinates representation, kinematics is described as in equations (2, 3, 4):

$$\dot{l} = k_1 l cos^2(\zeta), \tag{2}$$

$$\dot{\zeta} = -k_2\zeta - k_1 q_2 cos(\zeta)\frac{sin(\zeta)}{\zeta}, \tag{3}$$

$$\dot{\psi} = k_1 cos(\zeta) sin(\zeta), \tag{4}$$

where l is the distance between the desired and the initial position, ψ is the desired angle, and ζ is the angle between the desired position and the initial position, minus the angle of the initial orientation. Therefore $\dot{l}$, $\dot{\psi}$, $\dot{\zeta}$ represent the speed of variables l, ψ, ζ respectively [16-17].

Although this control law satisfies the stability condition, control can lead to undesirable conditions in some tasks, such as: i). the distance from the initial position to the final position must be greater than zero; and ii).

The asymptotic condition causes the robot to start with a very high speed and end up at a very slow speed, and iii) when the desired pose is not placed in front of the robot, the generated trajectory traces sharp curves, or takes the robot in reverse (v is negative) [17].

In [18], the control law used for the pose control of Robotino is based on a PI controller as follows equation (5):

$$\begin{bmatrix} v_{xc} \\ v_{yc} \\ v_{ac} \end{bmatrix} v_{c1} = -k_p * \begin{bmatrix} (x_d - x) \\ (y_d - y) \\ (\phi_d - \phi) \end{bmatrix} - ki \int \begin{bmatrix} (x_d - x) \\ (y_d - y) \\ (\phi_d - \phi) \end{bmatrix}. \tag{5}$$

This control law presents undesirable conditions as in the previous case of the unicycle robot. In [4], the generated trajectories for cases where the initial pose of the robot was $[0, 0, 0]$, and the different desired poses are the follows: a) $[3.5, 2.5, 0]$, b) $[3.5, 2.5, 20]$, c) $[3.5, 2.5, 40]$, d) $[3.5, 2.5, 60]$ e) $[3.5, 2.5, 80]$ was studied. It was observed, that if the desired angle has a higher value, the curvature generated is greater.

To face this problem, fuzzy-PI controllers were implemented to smooth the trajectories. However, trajectories were still sensitive to the value of the desired orientation. The following section presents the proposed three-step control strategy and the control law used for pose control, which attempts to overcome this design problem.

4 Three-step Position Control

This section presents a three-step strategy for controlling desired pose. Given a desired pose $pos_d = [x_d, y_d, \phi_d]$, and an initial position of the robot $pos_0 = [x_0, y_0, \phi_0]$, the entire task is divided into three steps as follows.

1. A steering angle ϕ_s is determined, with the aim of driving the robot front to the desired position x_d, y_d. The angle ϕ_s is determined as it is indicated in the equation (6):

$$\phi_s = atan2(y_d - y, x_d - x), \tag{6}$$

where $atan2$ is a function that extends to the tangent mathematical function, and generates angles $\phi_s \in (-\pi, \pi]$. Once the final position is determined, the control law to angular position can be applied over the action variable v_a. The control objective is $\phi_0 \rightarrow \phi_s$.

2. Once the robot has been driven toward the position ϕ_s, then a control position law can be applied over the action variable v_x, by defining the distance between $pos_d = [x_d, y_d]$ and $pos_0 = [x_0, y_0]$ as in equation (7):

$$d = \sqrt{(x_d - x)^2 + (y_d - y)^2}). \tag{7}$$

The control objective is $d \rightarrow 0$.

3. Once the robot is in the desired position, and it is facing over the orientation ϕ_s. Then the angular position control is applied again. Now, the control objective is $\phi_s \rightarrow \phi_d$.

By sequentially executing these three steps, the goal of pose control is achieved, and the drawbacks discussed in the previous section are avoided. From the presented procedure, it can be seen that the system needs two controllers. One to

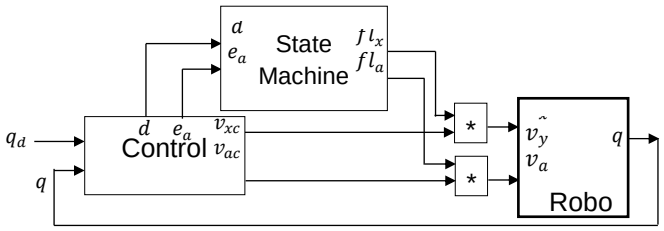


Fig. 2. Three-step position control schematic

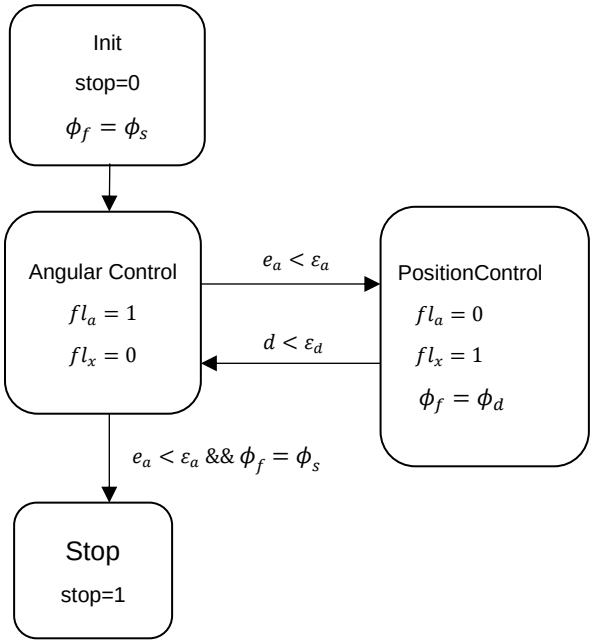


Fig. 3. Three-step pose control status machine

perform angular position control, to control the variable v_a, and another to perform position control to the variable v_x.

Control System Scheme

The control system schematic for the three-step pose control, which uses two controllers is shown in Figure 2. The action variables v_x, and v_a can be controlled by classic PID control or through fuzzy logic controllers, among others.

This document presents an implementation with a proportional configuration, due to its simplicity and efficiency. The control law implemented in the Control block is described in equations (8), and (9):

$$v_{xc} = k_{px} d, \tag{8}$$

$$v_{ac} = k_{pa}(\phi_d - \phi). \tag{9}$$

The goal of the state machine block is to switch the operating periods of the robot's action variables, and by using the flags fl_x and fl_a, and multipliers, the values of v_x and v_a can be turned-on or turned-off. Figure 3 shows the state machine logic of the proposed pose control. In the Init block, the stop flag is initialized at zero, and $\phi_f = \phi_s$, according to equation (6).

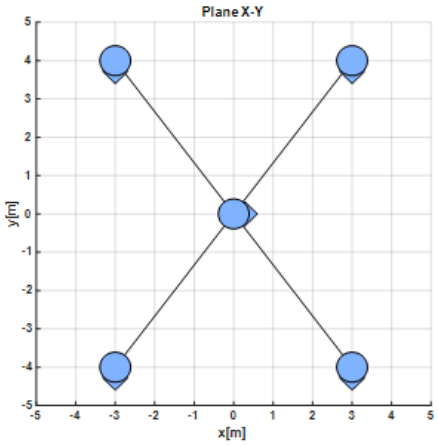


Fig. 4. Four desired poses that the robot should achieve with the initial position $q_0 = [0,0,0]$

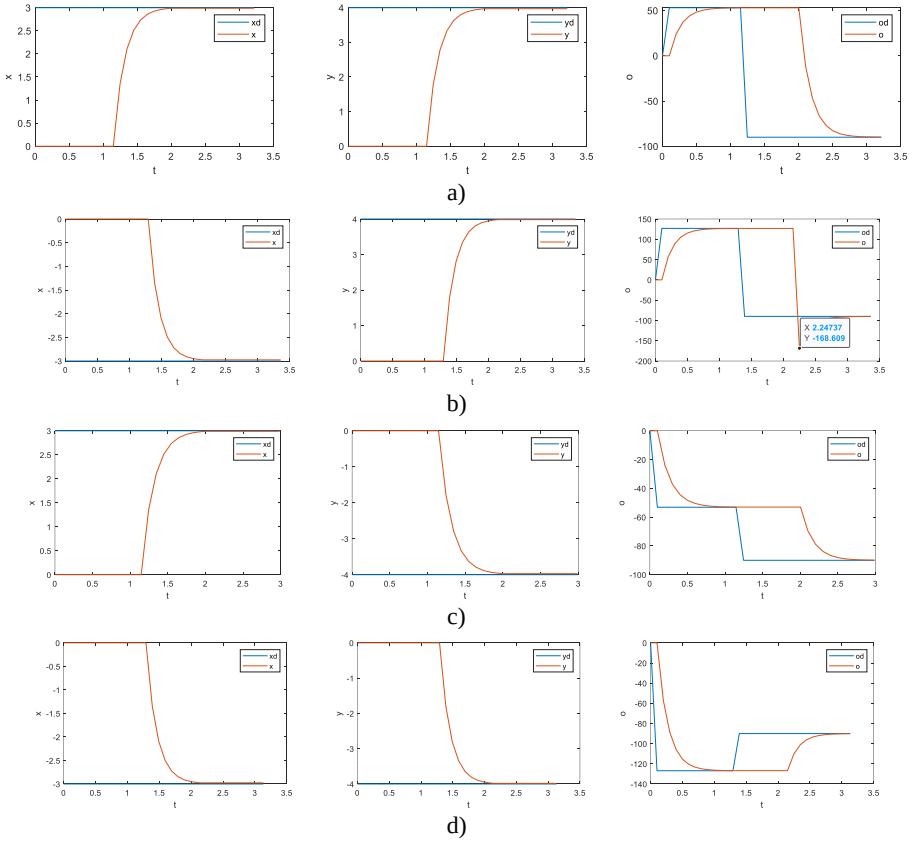


Fig. 5. Trajectories generated by the robot for the different desired poses in example 1

Once these variables are initialized, the Angular Control block is executed which sets $fl_a = 1$. This block remains active as long as $(\phi_f - \phi)$ is greater than a very small predefined value ε_a. When $e_a <$

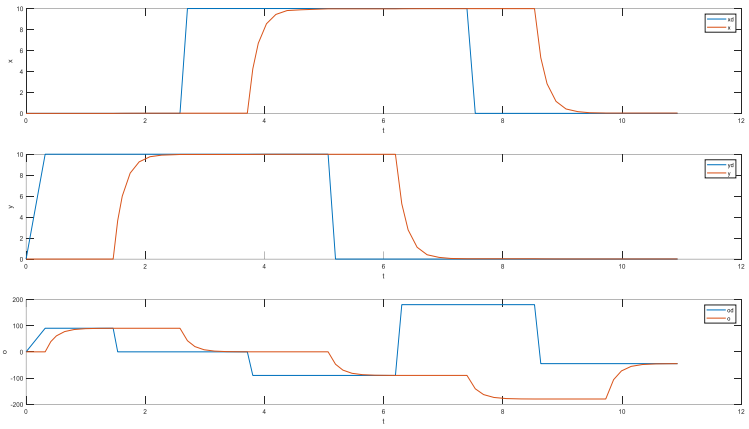


Fig. 6. Square closed-loop path traveled by the robot

ε_a is met, the PositionControl block is executed and the variables are set as shown in the figure. The system will exit of this state when the robot is in the desired position, i.e. until $d < \varepsilon_d$.

Note the assignment of $\phi_f = \phi_d$ made in this block, changes the conditions when system returning to the Angular Control state, and the robot could be oriented towards ϕ_d. Once $e_a < \varepsilon_a$ is met again, and now the system complies $\phi_f = \phi_s$, the state machine is directed to the Stop state, where the stop flag is turn on to end the pose control process.

Finally, three-step pose control can be applied to autonomous navigation problems. Assuming that you have a model of the environment, and the path for the robot to reach its goal is given through a sequence of reference points. Then, given an initial pose $pos_0 = [x_0, y_0, \phi_0]$ of the robot, and a sequence of positions that the robot must pass in the xy-plane $p_d = [(x_1, y_1), (x_2, y_2), \dots, (x_f, y_f)]$, and a final orientation ϕ_f, the three-step rule described above can be consecutively applied to the robot will reach its goals through on the xy- plane.

Note that the proposed strategy only focuses on the behavior of action on the robot's motors, and the obstacle avoidance and/or the automatic construction of the sequence of points of the path are not within the scope of this research work.

5 Experimental Results

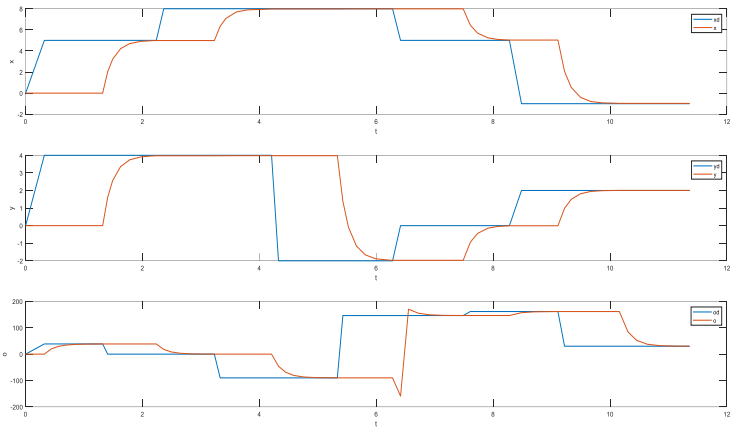
The operation of the proposed control strategy will be shown in some cases. The first case shows the pose control for different situations where the robot is driven to different points over the Cartesian plane, with a robot's initial position in the center of the plane, and looking towards the x-axis.

The second case shows the operation of generating and tracking trajectory, and given a series of desired positions the robot is required to pass. In example 2, the robot will be traveling deploying a square path given 3 points. In example 3, the robot should travel an open circuit given a set of desired points. In all cases, the gains of the controllers are: $k_{px} = 2$, and $k_{pa} = 6$.

Example 1. Pose control for an omnidirectional robot with an initial position at the origin and orientation zero, and having as its destination different final positions that lead to the different quadrants of the Cartesian plane.

So, given the initial configuration $q_0 = [0,0,0]$, and the desired configurations: a) q_{d1}= [3,4,-90]; b) q_{d2}= [-3,4,-90]; c) q_{d3}= [3,-4,-90]; d) q_{d4}= [-3,-4,-90], as shown in Figure 4. Note that the final orientation in all cases is $\phi_d = -90$, however, for the robot, it is a different challenge due to its steering angle ϕ_s is different in all cases.

Figure 5 shows plots to indicate the path taken by the different variables of the robot on the xy-plane as a time function. For case a) it is observed



.

Fig. 7. Open circuit traveled by the robot in example 3

that the ϕ_s calculated is 53.13 degrees, for the rest of the cases, ϕ_s is 143.13, -36.87, -126.87 respectively, and it was reached in approximately 1.2s. It can be noticed that the robot starts its translation movement after this time, which is 1.2s, and ends in about 2s. The robot continues with the final rotational movement that leads it to $\phi_d = -90$.

Example 2. The robot travels through a square closed circuit of 10 m^2.

The robot starts in its initial pose $q_0 = [0,0,0]$, and the sequence of desired points and the predefined final orientation are $p_d = [(0\ 10), (10\ 10), (10, 0), (0,0)]$, $\phi_f = -45$, respectively. Note that the final desired position, $(0,0)$ is obtained from the initial position of the robot. Figure 6 shows the trajectory generated by the robot.

The navigation path has only the positions where the robot should pass, therefore the orientations are calculated using the pose control rule. The orientation angles obtained by the method are: $\phi_s = [90, 0, -90, 180]$, as are shown in Figure 6. In addition, it can be seen that the pose control rule was repeated 4 times, and the final position was reached in approximately 11s.

In the angular variable plot, for a time around 8s, the $\phi_s = 180$, and the plot indicates that the robot reached $\phi_s = -180$, which is equivalent to a zero-degree error.

Example 3. The robot should travel an arbitrarily and predefined circuit based on the definition of a sequence of desired points, and the definition of a final orientation. Let and $q_0 = [0,0,0]$, and $p_d = [(5\ 4), (8\ 4), (8, -2), (5,0), (-1,2)]$, and $\phi_f = 30$.

Figure 7 shows the trajectory generated by the robot. The angles calculated in the three-step pose control strategy are: $\phi_s = [38.66, 0, -90, 146.31, 161.56]$. Notice that when t~6.3, the angle goes from -90 to 146.31 degrees, the robot displayed an undesired behavior, however, the robot corrects this and achieves the goal.

The above examples have shown how the pose control strategy works. By sequentially repeating the three-step pose control rule, the proposal was applied to navigation problems. For simplicity, the position controllers used in the proposed strategy are implemented with proportional controllers, and for the kinematic model, it was observed that they are very efficient. However, in real robotics platforms, it is likely to require robust controllers such as PID control or fuzzy logic, among others.

On the other hand, one of the main objectives of the proposal is to avoid the non-smooth curves that are generated by some of the most widely used pose controllers found in the literature. In this work, it was reached by controlling only the variables v_x and v_a of the omnidirectional robot.

However, for increasing the robustness of the proposed rule, it could control v_x and v_y at the same time in step 2, and retain the desired

behavior, which is that the translation movement of the robot is always over the front view of the robot.

Note the behavior generated by the proposed three-step control strategy, can be implemented in a different type of robots, for example in a differential drive robot, or even in drones, by performing yaw movements in steps 1 and 3, and roll and pitch movements in step 2.

Finally, a comparison of the proposed method with other methods oriented to pose control is difficult, because all methods found by the authors of this work are addressed in achieving the objective in the shortest possible time, in contrast to the objective of the proposal of this work which is that the translation movement is performed in the direction of the front of the robot.

Intuitively it is expected that this method takes a little longer to perform the pose control task since this proposal performs it in three steps.

Therefore, as future work and/or as an exercise the reader is invited to perform the examples shown in this section, and compare the behavior of the robot in the generation of the trajectories, time consumed, and error rates when using the presented method and the control used in equations (2), (3), and (4) or in other methods that could be found in the literature.

6 Conclusions and Future Work

In this article, a control strategy for the position and orientation control of an omnidirectional mobile autonomous robot, was presented and carried out in three steps. Using the kinematic model of an omnidirectional robot, it was shown by different case studies that the robot meets the objective of pose control, and it was observed in all cases that the robot did not describe undesirable sharp curves.

In future work, the task of avoiding obstacles will be added to the proposed strategy and will be implemented on the Robotino platform to verify its behavior in real situations. On the other hand, fuzzy controllers will be added to the strategy, and it will be applied (with some modifications) for the pose control of a differential drive robot and an unmanned aerial robot.

References

1. **Kebede, G.A., Gelaw, A.A., Andualem, H., Hailu, A.T. (2024).** Review of the characteristics of mobile robots for health care application. International Journal of Intelligent Robotics and Applications, pp. 1–23. DOI: 10.1007/s41315-024-00324-3.
2. **Rubio, F., Valero, F., Llopis-Albert, C. (2019).** A review of mobile robots: Concepts, methods, theoretical framework, and applications. International Journal of Advanced Robotic Systems, Vol. 16, No. 2, DOI: 10.1177/17298814198395.
3. **Hong, T.S., Nakhaeinia, D., Karasfi, B. (2012).** Application of fuzzy logic in mobile robot navigation. Fuzzy Logic-Controls, Concepts, Theories and Applications, pp. 21– 36.
4. **Sousa, R.B., Costa, P.G., Moreira, A.P. (2021).** A pose control algorithm for omnidirectional robots. IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 91–96. DOI: 10.1109/ICARSC522 12.2021.9429803.
5. **Figueroa-Saire, P., Patiño-Escarcina, R. (2021).** Position control of an omnidirectional robot through visual servoing. Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE) IEEE, pp. 258–263. DOI: 10.1109/LARS/SBR/WRE54079.2021.9605393.
6. **Luo, D., Schauer, T., Roth, M., Raisch, J. (2012).** Position and orientation control of an omni-directional mobile rehabilitation robot. IEEE International Conference on Control Applications, pp. 50–56. DOI: 10.1109/CCA.2012.6402680.
7. **Varlamov, O. (2021).** Brains for robots: Application of the Mivar expert systems for implementation of autonomous intelligent robots. Big Data Research, Vol. 25, DOI: 10.1016/j.bdr.2021.100241.
8. **Rana, K., Talbot, B., Dasagi, V., Milford, M., Sünderhauf, N. (2020).** Residual reactive navigation: Combining classical and learned navigation strategies for deployment in

unknown environments. IEEE International Conference on Robotics and Automation, pp. 11493–11499. DOI: 10.1109/ICRA40945. 2020. 9197386.

9. **Cuevas, F., Castillo, O., Cortes, P. (2022).** Optimal setting of membership functions for interval type-2 fuzzy tracking controllers using a shark smell metaheuristic algorithm. International Journal of Fuzzy Systems, Vol. 24, No. 2, pp. 799–822. DOI: 10.1007/s40815-021-01136-4.
10. **Wang, D., Tan, D., Liu, L. (2018).** Particle swarm optimization algorithm: An overview. Soft computing, Vol. 22, No. 2, pp. 387–408. DOI: 10.1007/s00500-016-2474-6.
11. **Ma, Y., Wang, Z., Yang, H., Yang, L. (2020).** Artificial intelligence applications in the development of autonomous vehicles: A survey. IEEE/CAA Journal of Automatica Sinica, Vol. 7, No. 2, pp. 315–329. DOI: 10.110 9/JAS.2020.1003021.
12. **Kunze, L., Hawes, N., Duckett, T., Hanheide, M., Krajník, T. (2018).** Artificial intelligence for long-term robot autonomy: A survey. IEEE Robotics and Automation Letters, Vol. 3, No. 4, pp. 4023–4030. DOI: 10.1109/LRA.2018. 2860628.
13. **Lynch, K.M., Park, F.C. (2017).** Modern robotics. Cambridge University Press.
14. **Castillo, O., Cortés-Antonio, P., Melin, P., Valdez, F. (2020).** Type-2 fuzzy control for line following using line detection images. Journal of Intelligent & Fuzzy Systems, Vol. 39, No. 5, pp. 6089–6097. DOI: 10.3233/JIFS-189081.
15. **González-Aguilar, J., Castillo, O., Cortés-Antonio, P. (2019).** Implementation a fuzzy system for trajectory tracking of an omnidirectional mobile autonomous robot. Hybrid Intelligent Systems in Control, Pattern Recognition and Medicine, Cham: Springer International Publishing, Vol. 827, pp. 327–340. DOI: 10.1007/978-3-030-34135-0_23.
16. **Tzafestas, S.G. (2018).** Mobile robot control and navigation: A global overview. Journal of Intelligent & Robotic Systems, Vol. 91, pp. 35–58. DOI: 10.1007/s10846-018-0805-9.
17. **Corke, P. I., Jachimczyk, W., Pillat, R. (2011).** Robotics, vision and control: fundamental algorithms in MATLAB. Springer, Vol. 73.
18. **Masmoudi, M.S., Krichen, N., Masmoudi, M., Derbel, N. (2016).** Fuzzy logic controllers design for omnidirectional mobile robot navigation. Applied soft computing, Vol. 49, pp. 901–919. DOI: 10.1016/j.asoc.2016. 08.057.

Article received on 17/06/2024; accepted on 15/08/2024.
**Corresponding author is Prometeo Cortés-Antonio.*