Noname manuscript No.
(will be inserted by the editor)

A Feed Forward Trajectory Control Algorithm for Omnidirectional Robots

Ricardo B. Sousa[1,2] · **Paulo G. Costa**[1,2] · **Héber Miguel Sobreira**[2] · **António Paulo Moreira**[1,2]

Received: date / Accepted: date

Abstract The pose control (position and orientation) of omnidirectional robots are of great interest due to their complete maneuverability being suitable for dynamic environments. So, we propose a pose control algorithm for omnidirectional robots using Proportional-Integrative (PI) controllers for angular speed control of the wheels' motors, and Proportional-Derivative (PD) and Feed-Forward (FF) controllers to control the pose of the robot in space and in time. The algorithm uses polynomial approximations of first and second-order to compute the derivatives needed for the FF controllers considering a subset of points (future trajectory) of the desired trajectory. Experiments in a simulation environment and with a three-wheeled omnidirectional soccer robot were performed to analyze the optimal size of the future trajectory, compare the proposed algorithm with a PD-only pose control scheme, and evaluate the processing time usage of the proposed controller. The results demonstrated that the proposed controller was better than a PD-only pose control scheme being suitable for pose control of omnidirectional robots and for systems with fast dynamics or focused on low computation resources (due to the low processing time usage). Also, it was formulated how to define the size of the future trajectory depending on the current reference velocity of the robot based on the experimental results.

Keywords Control · Omnidirectional robots · Pose control · Motion control

This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020. This work is also financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within scholarship 2021.04591.BD.

Ricardo B. Sousa
up201503004@edu.fe.up.pt

Paulo G. Costa
paco@fe.up.pt

Héber Miguel Sobreira
heber.m.sobreira@inesctec.pt

António Paulo Moreira
amoreira@fe.up.pt

[1] Faculty of Engineering of the University of Porto, Electrical Engineering Department, Porto, 4200-465 Porto, Portugal
[2] INESC TEC – Institute for Systems and Computer Engineering, Technology and Science, CRIIS – Centre for Robotics in Industry and Intelligent Systems, Porto, 4200-465 Porto, Portugal

1 Introduction

Omnidirectional robots are of great interest in ground mobile robots. Their complete maneuverability allows them to move in all directions at any time being suitable for dynamic environments. Indeed, omnidirectional robots do not have nonholonomic kinematic constraints; so, they are considered to be holonomic robots. A consequence of not having nonholonomic constraints is that the translation and rotation components of the omnidirectional steering geometry are independent of each other. Omnidirectional robots are capable of performing translation and rotational motions simultaneously. One popular use of omnidirectional robots is in robot soccer games of the RoboCup competition and, nowadays, this steering geometry is used more and more in industrial applications [18].

The control of omnidirectional robots is crucial to control the pose (position and orientation) of the robot. The robot must reach the desired pose, but also we should be able to control how and when the robot gets there; in other words, the desired trajectory. Several works were proposed characterizing the nonlinearity and uncertain factors of these robots. Some approaches use fuzzy control [1,2,6,12] due to being able to use the experience of human experts to model

the control rules. Abiyev et al. [2] designed a Proportional-Derivative (PD) based fuzzy control system for the position and orientation angle to control linear and angular motions of a four-wheeled omnidirectional robot. This fuzzy control system had 7 memberships and 49 fuzzy rules. Similar to [2], Abiyev et al. [1] designed a PD-based fuzzy controller with 49 rules but used Z-number-based fuzzy interpolative reasoning for defining the control rules. Huang et al. [6] implemented a fuzzy control system to adjust online the parameters of a PD-based motion controller for three-wheeled omnidirectional robots. Masmoudi et al. [12] used two fuzzy adaptive tuners with conventional Proportional-Integrative (PI) controllers for controlling three-wheeled omnidirectional robots. A third fuzzy controller was used for obstacle avoidance. Other approaches used adaptive controllers in conjunction with neural networks [3, 10, 17]. Bugeja et al. [3] used neural networks for online function approximation to estimate the dynamic model of a differential robot and dual adaptive controllers for computing the torques required at the wheels (based on their angular speed references). Similar to [3], Rossomando and Soria [17] used a neural network for determining the dynamic model of a differential robot. An adaptive neural PID controller was implemented to compensate for the dynamic nonlinearities and the variations of the robot model's parameters. Li et al. [10] combined tracking control (using a neural network to deal with unmodeled extra disturbances and unstructured dynamics) with a PD controller and a robust control component for differential robots. Sliding mode control [7, 13] is also found in the literature studying the tracking problem as a stabilization one. Mu et al. [13] defined nonlinear sliding surfaces for the corresponding sliding mode dynamics of a differential mobile robot. As for Jianbin and Jianping [7], their approach was to implement an adaptive sliding mode control based on the Lyapunov function considering input constraints, model uncertainties, and external disturbances for a four-wheeled omnidirectional robot. Trajectory linearization control was another type of approach used by Liu et al. [11] to achieve robust stability and performance along the trajectory using also a low-pass filter for the trajectory commands. Although the low-pass filter smoothed abrupt changes in the trajectory commands, a delay was noticed between the commands and the response. Recently, assuming a known dynamic model of the robot, Model Predictive Control (MPC) [9, 20] is being used in trajectory control with the possibility of formulating constraints for the system. The work of Li et al. [9] incorporated a neural-dynamic optimization (used to solve the MPC quadratic programming problem) to achieve trajectory tracking of nonholonomic mobile robots (e.g., differential robots). Wang et al. [20] implemented an MPC algorithm with control and system constraints to achieve point stabilization and trajectory tracking. In contrast, other more classical approaches are based on the use of PI, PD, and/or controllers. For example, Watanabe et al. [21] determine a dynamic model of an omnidirectional robot to design an acceleration control scheme with PI and PD controllers.

However, approaches such as fuzzy control, adaptive control, neural networks, or sliding mode control that consider nonlinearities increase the complexity of the controller's implementation. Also, tunning fuzzy or neural networks-based controllers to achieve a certain closed-loop performance (e.g., settling time or overshoot) is sometimes not easy or clear. Real-time implementations of MPC algorithms (i.e., online optimization) require significant computation resources being not feasible for faster dynamic systems [9]. As for classical approaches that use PI and/or PD controllers, the overshoot depends on the robot's dynamics when sudden changes happen in the position reference, and it is possible to exist a delay between the reference and actual pose of the robot.

Therefore, this work is intended for controlling the pose of omnidirectional soccer robots with fast dynamics given a set of points in space with timestamps (position, orientation, and time) associated with each one. Conventional PI controllers with dead zone compensation of the motors are implemented to control the angular speed of the wheels. Based on experiments made with a three-wheeled omnidirectional robot, the robot's velocity model can be approximated by a first-order system. So, PD controllers are used for positioning control, and Feed-Forward (FF) controllers mitigate the delay between the robot's reference and actual pose by inverting the robot's velocity model. This control scheme allowed us to occupy less than 1.5% of the 40ms pose control period, as shown in the experiments. Also, the pose control uses a subset of points from the desired trajectory as future poses of the robot to compute the first and second derivatives required for the FF controllers. Simulation and real experiments with a three-wheeled omnidirectional soccer robot show that this approach of computing the derivatives leads to not only the robot following the trajectory on the desired time instants without any delay while predicting sudden changes in the pose's reference.

Our previous work [19] is the basis for the formulation of the trajectory controller. This article extends that work by defining a methodology to characterize and compensate the motors' dead zone, how to obtain the robot's velocity model, and how the future trajectory should be built. Also, this work presents tests with a real robot including an analysis of the computation resources used by the proposed controller and the impact of the future trajectory's size on the tracking accuracy and computation resources.

The article is organized as follows. Section 2 defines the considerations and desirable characteristics when defining a trajectory for the robot. Section 3 formulates a classical approach to pose control using only PI and PD controllers and presents methodologies to define the parameters of these controllers, characterize and compensate the motors' dead

zone, and retrieve the robot's velocity model. Section 4 proposes the use of FF controllers and defines the formulation of the controllers' derivatives and the subset of points that represent the future trajectory. Section 5 analyses the results obtained from the experiments made with a simulated and a real three-wheeled omnidirectional robot. Finally, Section 6 presents the conclusions and future work.

2 Trajectory Definition

2.1 Type of trajectory

Usually, a planner defines a trajectory as parametric splines, arcs, or a set of points. The first two definitions can be discretized in time to obtain a set of points with timestamps associated with each point. So, a set of points can describe any type of trajectory that a planner is using. In this article, we considered a trajectory T as a set of points $P_{r,k}$ defined both in space (position and orientation of the robot) and in time: $T = \{P_{r,0}(t_0), P_{r,1}(t_1), \ldots, P_{r,N-1}(t_{N-1})\}$, where $P_{r,k}(t_k) = \{X_{r,k}, Y_{r,k}, \theta_{r,k}\}(t_k)$.

The set of points that describe a trajectory can be defined in three different ways. The generic formulation is defining the points without any restrictions in space or in time. Two other definitions are restricting the distance (e.g., maintaining a certain distance) or time (e.g., constant time intervals) between consecutive points. The formulation used in this article is restricting the time interval as a constant value between consecutive points. As it is explained in Section 4, this definition has the advantage of reducing the computation resources required by the proposed trajectory controller. However, the proposed controller works also with the two other ways of defining the set of points.

2.2 Coordinate frame

As for the reference coordinate frame of the trajectory, it can be relative to the world's coordinate frame ($\{X_r^W, Y_r^W, \theta_r^W\}$) or the robot's local frame ($\{X_r^R, Y_r^R, \theta_r^R\}$). These two alternatives are illustrated in figure 1 for the point $P_{r,k}$ of a certain trajectory T, where $P_{r,k}^W$ represents the point's desired pose relative to the world and $P_{r,k}^R$ to the robot. The orientation matrix $R^{-1}(\theta_{rb}^W)$ defined in equation 1 (where $\cos\alpha$ and $\sin\alpha$ of a certain angle α are represented by c_α and s_α, respectively) is the orientation of the world relative to the robot's local coordinate frame depending on the robot's current orientation (θ_{rb}^W). The transformation of the trajectory from the world to the robot's local frame can be defined as in equation 2.

$$R^{-1}(\theta_{rb}^W) = \begin{bmatrix} c_{\theta_{rb}^W} & s_{\theta_{rb}^W} & 0 \\ -s_{\theta_{rb}^W} & c_{\theta_{rb}^W} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

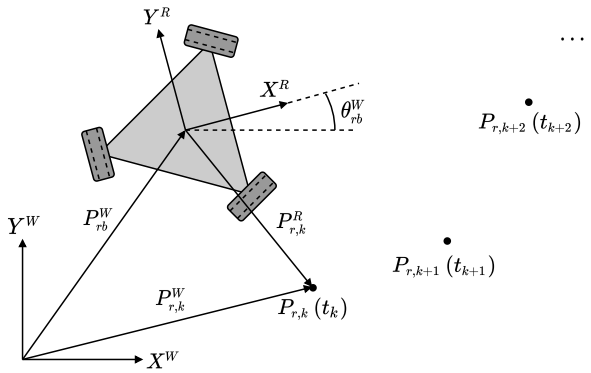


Fig. 1 Trajectory relative to the world and robot's coordinate frames

$$\begin{aligned} \begin{bmatrix} X_{r,k}^R \\ Y_{r,k}^R \\ \theta_{r,k}^R \end{bmatrix} &= R^{-1}(\theta_{rb}^W) \cdot \left[\begin{bmatrix} X_{r,k}^W \\ Y_{r,k}^W \\ \theta_{r,k}^W \end{bmatrix} - \begin{bmatrix} X_{rb}^W \\ Y_{rb}^W \\ \theta_{rb}^W \end{bmatrix} \right] \\ &= \begin{bmatrix} (X_{r,k}^W - X_{rb}^W)\cdot c_{\theta_{rb}^W} + (Y_{r,k}^W - Y_{rb}^W)\cdot s_{\theta_{rb}^W} \\ (-X_{r,k}^W + X_{rb}^W)\cdot s_{\theta_{rb}^W} + (Y_{r,k}^W - Y_{rb}^W)\cdot c_{\theta_{rb}^W} \\ \theta_{r,k}^W - \theta_{rb}^W \end{bmatrix} \end{aligned} \tag{2}$$

2.3 Desirable characteristics

Lastly, the trajectory generation should have some desired characteristics to avoid saturation on the controller. Only one requirement was followed in this article: a continuous time evolution for the module of the robot's linear velocity (v). In figure 2, it is shown a squared trajectory considering an initial acceleration and final deceleration of 1m.s^{-2}, a nominal velocity of 1m.s^{-1}, and a final velocity approximation to the desired goal of 0.1m.s^{-1}. As illustrated in figure 3, the requirement considered in this article removes any step changes on the robot's linear velocity (and, consequently, on its desired position X_r^W and Y_r^W) and orientation (θ_r^W).

Although it is desired a continuous first (velocity) and second (acceleration) derivatives of the generated trajectory (both in module and direction), the trajectories used in the experiments only assumed that the planner generates at least a continuous time evolution for the module of the robot's linear velocity. Works such as Petrinec and Kovacic [15] or Klančar et al. [8] should be consider to assure the continuity of the first and second derivatives.

3 A Classical Approach to Pose Control

In this section, it is proposed a pose control scheme for omnidirectional robots (the three-wheeled steering geometry used in the experiments is illustrated in figure 4) using conventional PI controllers for the angular speed of the wheels

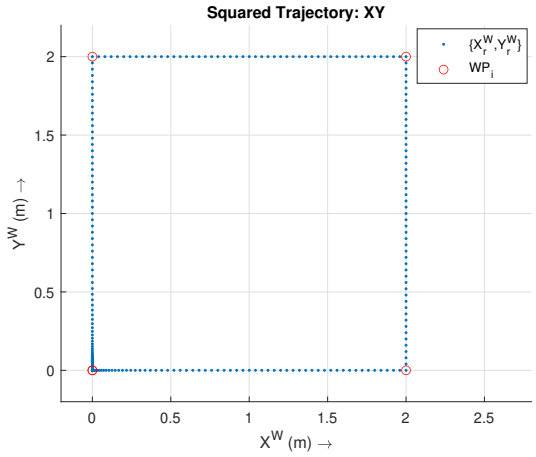


Fig. 2 Squared trajectory with continuous linear velocity's module – global coordinate frame $\{X^W, Y^W\}$

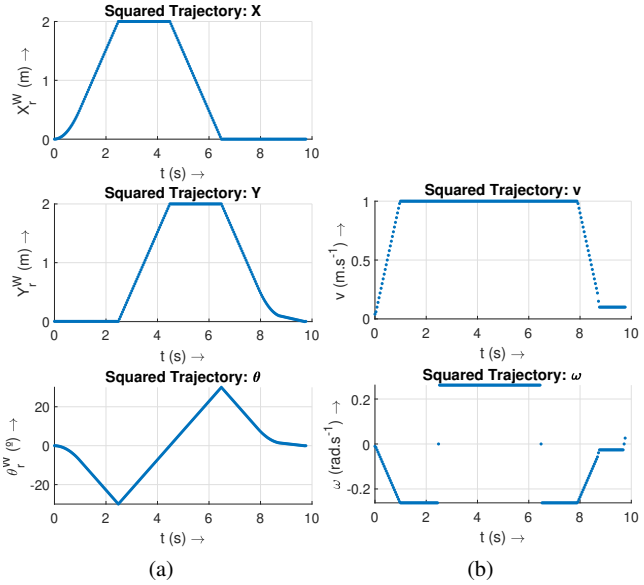


Fig. 3 Squared trajectory with continuous linear velocity's module – analysis over time: (a) position $\{X^W, Y^W, \theta^W\}$; (b) velocity $\{v, \omega\}$

and conventional PD controllers for the robot's pose relative to its own coordinate frame.

3.1 Inverse differential kinematics

First, it is necessary to define the inverse differential kinematics of omnidirectional robots to control their velocity through a specific trajectory. Given the linear (v and v_n along the directions of X^R and Y^R, respectively) and angular (ω) velocities desired for the robot, equation 3 computes the linear velocity (v_i) of each wheel i (considering also the distance l between the robot's geometric center and the wheels) [18]. Then, the linear velocity (v_i) of each wheel is converted into angular speed ($\dot{\varphi}_i$) depending on the diameter of the wheels (D_i where $i = 1,2,3$), as illustrated in equation 4.

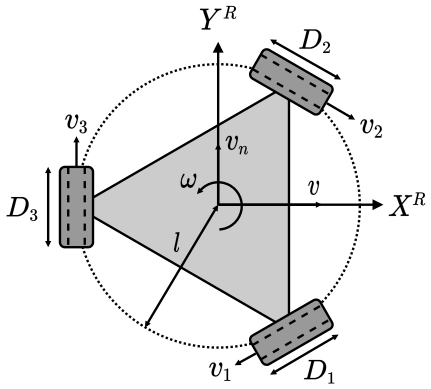


Fig. 4 Three-wheeled omnidirectional robot with its local coordinate frame $\{X^R, Y^R\}$ and kinematic parameters (l: distance between the robot's geometric center and the wheels; D_i: diameter of the wheel i where $i = 1,2,3$)

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} -\frac{\sqrt{3}}{2} & -\frac{1}{2} & -l \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} & -l \\ 0 & 1 & -l \end{bmatrix} \cdot \begin{bmatrix} v \\ v_n \\ \omega \end{bmatrix} \tag{3}$$

$$\dot{\varphi}_i = \frac{2}{D_i} \cdot v_i \tag{4}$$

3.2 Angular speed control of the wheels

Next, the angular speed of the wheels ($\dot{\varphi}_i$) can be controlled using conventional PI controllers to set the motors' input voltage (V_i). The PI controllers were tuned using the Internal Model Control (IMC) method [4]. Given experiments performed with a real robot in the ground, the system can be considered a first-order one. So, [4] requires the output gain ($K_{\dot{\varphi},p}$), the time constant ($\tau_{\dot{\varphi}}$), and the lag ($L_{\dot{\varphi}}$) to be estimated. The PI parameters (the proportional gain $K_{\dot{\varphi},c}$ and integration time $T_{\dot{\varphi},I}$) are computed using equation 5, given a desired time constant for the closed-loop ($\tau_{\dot{\varphi},cl}$).

$$\begin{cases} K_{\dot{\varphi},c} = \frac{1}{K_{\dot{\varphi},p}} \cdot \frac{\tau_{\dot{\varphi}}}{\tau_{\dot{\varphi},cl} + L_{\dot{\varphi}}} \\ T_{\dot{\varphi},I} = \tau_{\dot{\varphi}} \end{cases} \tag{5}$$

Consequently, the responses of the angular speed of the wheels were analyzed to validate our consideration of these responses as a first-order system. Although it was noted in the experiments that the response was slower when the robot started from a standstill pose relative to already being in motion, we only analyzed the later one for simplification purposes. Figure 5 presents three different responses of the wheels' angular speed respective to three different types of motion: v, v_n, and ω. Using an optimization algorithm that minimizes the difference between a first-order model and the

Table 1 Experimental results relative to retrieving model's parameters of the 1st-order approximation of the $\dot{\varphi}_i = f(V_i)$ responses

	$K_{\dot{\varphi},p}$ (rad.s^{-1}/V)			$\tau_{\dot{\varphi}}$ (s)		
	M1	**M2**	**M3**	**M1**	**M2**	**M3**
v	2.7359	2.7579	–	0.2395	0.2889	–
	2.7457	2.5793	–	0.2450	0.2677	–
	2.7061	2.7300	–	0.2703	0.3067	–
	2.6986	2.7332	–	0.2209	0.2704	–
	2.8147	2.8160	–	0.2262	0.2284	–
v_n	2.5639	2.6152	2.6757	0.2902	0.2816	0.2170
	2.4270	2.6678	2.5827	0.2510	0.2779	0.2225
	2.5908	2.8199	2.5917	0.2368	0.2366	0.2486
	2.5332	2.7851	2.5560	0.2172	0.2661	0.2513
ω	2.7920	2.9490	2.8873	0.1130	0.1283	0.1326
	2.9664	2.8491	2.7588	0.1184	0.1187	0.1142
	2.9901	2.6673	2.7909	0.1116	0.0928	0.1064
	3.0285	2.9516	2.7268	0.1494	0.1147	0.1410

real response, we retrieved the parameters of the first-order approximation ($K_{\dot{\varphi},p}$ and $\tau_{\dot{\varphi}}$). The responses shown in figure 5 are very similar (first-order approximation versus real response) with an absolute mean error of 2.31%, 2.71%, and 2.74% (relative to the steady-state angular speed) for the 1st-order approximations of the motions v, v_n, and ω, respectively. So, we validated that the angular speed responses of all wheels and types of independent motions can be considered a first-order system for the robot used in the experiments (see Section 5). Several experiments were performed and the results are presented in table 1. Although the systems are clearly different for the angular motion ω versus the two other types of motions (with a mean for $\tau_{\dot{\varphi}}$ of 0.1201s being approximately two times lower than 0.2564s and 0.2497s for v or v_n, respectively), we used the average of all experiments to retrieve the parameters $K_{\dot{\varphi},p}$ and $\tau_{\dot{\varphi}}$, for simplification purposes. The PI parameters are computed as in equation 5, and their values are the following ones (considering a $\tau_{\dot{\varphi},cl} = \tau_{\dot{\varphi}}/1.5$ to decrease in 66.67% the response time):

$$\begin{matrix} K_{\dot{\varphi},p} & = 2.7374 \\ \tau_{\dot{\varphi}} & = 0.2087\text{s} \end{matrix} \rightarrow \tau_{\dot{\varphi},cl} = 0.1392\text{s} \rightarrow \begin{matrix} K_{\dot{\varphi},c} & = 0.5480 \\ T_{\dot{\varphi},I} & = 0.2087\text{s} \end{matrix}$$

Considering that the motors have a dead zone – nonlinearity due to static frictions –, a Hammerstein nonlinear block was used to reduce the influence of this zone. The main goal of the Hammerstein nonlinear block is to reduce the motors' dead zone from V_0 (when the motors start rotating, without the nonlinear block) to a new zone V_d, as illustrated in figure 6. Although the Hammerstein is a nonlinear block, the response of the angular speed of the wheels in relation to the voltage applied to the motors is more linear with the Hammerstein block than without it. Also, V_d should not be 0 to be possible to stop the motors independently of the surface the robot is moving on and/or type of motion.

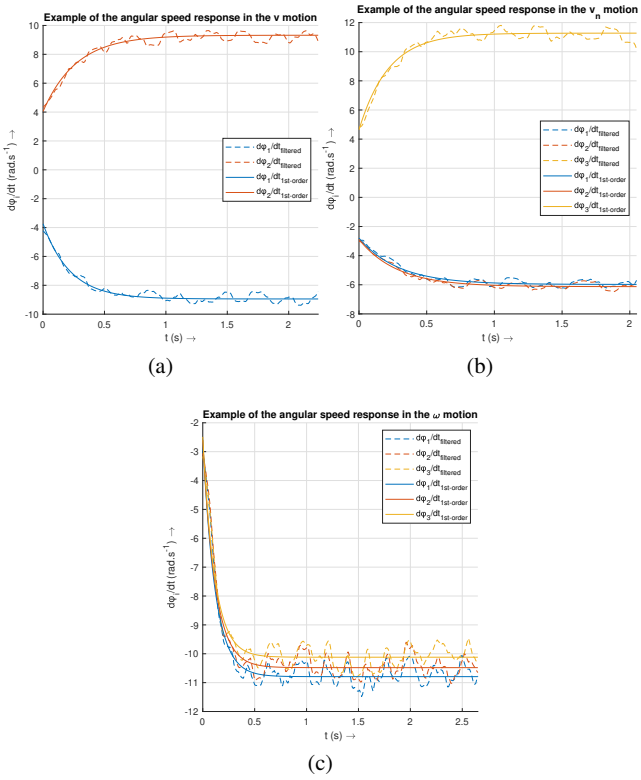


Fig. 5 Responses of the angular speed of the wheels ($\dot{\varphi}_i$) for the following types of motions: (a) v; (b) v_n; (c) ω

Algorithm 1: Hammerstein nonlinear block to compensate the dead zone of the motors

input : $V_{mot,i}$,$V_{d,i}$,$V_{0,i}$
output: V_i

if $V_{mot,i} > V_{d,i}$ **then**
  $V_i = (V_{mot,i} - V_{d,i}) + V_{0,i}$
else if $V_{mot,i} > -V_{d,i}$ **then**
  if $V_{d,i} \neq 0$ **then** $V_i = V_{mot,i} \cdot V_{0,i}/V_{d,i}$
  else $V_i = 0$
else
  $V_i = (V_{mot,i} + V_{d,i}) - V_{0,i}$

The Hammerstein nonlinear block used in this article is described in the algorithm 1.

Experiments were performed to define the parameters of the Hammerstein block. Given that the dead zone is due to static frictions that can vary slightly between similar tests, it was performed 10 experiments for each type of independent motion of an omnidirectional robot: v, v_n, and ω. The surface used was the same one on which the pose controller will be tested and used in real applications – a carpet of a soccer robot's field (see Section 5). Next, the voltage up to which the robot remained stopped in each type of motion was annotated for each experiment. The results are presented in table 2 illustrating the statistic measures for each v, v_n, and ω. Although v and v_n are very similar in terms

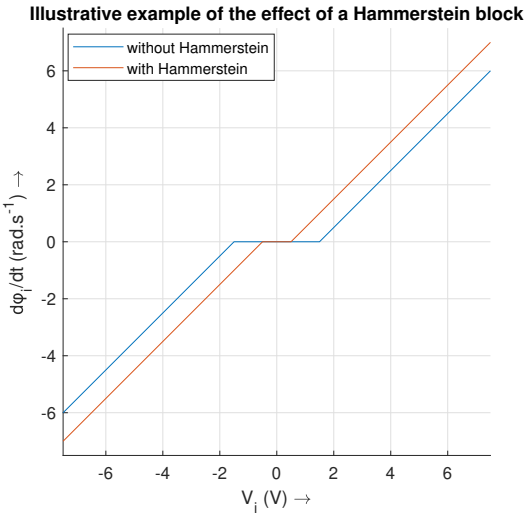


Fig. 6 Illustrative example of the nonlinear Hammerstein block's effect on the system $\dot{\varphi}_i = f(V_i)$ (with $V_{0,i}$ = 1.5V and $V_{d,i}$ = 0.5V)

Table 2 Statistic measures of the analysis of the motors' dead zone (μ: mean; σ: standard deviation; m: median; min: minimum; max: maximum)

(V)	v	v_n	ω
μ	3.4490	3.3955	1.6299
σ	0.2883	0.1390	0.1828
m	3.5162	3.5010	1.6288
min	2.7019	3.2317	1.3563
max	3.6581	3.5061	2.0418

of dead zone, the angular motion ω demonstrated to have an approximately 50% smaller zone than v or v_n (median of 1.6288V vs 3.5162V and 3.5010V, respectively). The main reason is that the angular motion ω leads to all wheels rotating in the same direction, while in v wheel 1 has an opposite direction of wheel 2 and in v_n wheel 3 has the opposite direction to wheels 1 and 2. In order to be able to stop the robot independently of the type of motion, $V_{0,i}$ should be lower than the minimum dead zone found for the different motions tested. So, $V_{0,i}$ for all motors was defined as 80% of 1.6288V (median of the dead zone for the angular motion ω), i.e., 1.3030V. This value set for $V_{0,i}$ was lower than all the values for the dead zone obtained in the experiments, as shown in table 2. $V_{d,i}$ was set as 0.5212V to reduce the dead zone in 60% relative to $V_{0,i}$.

3.3 Pose control

The evolution of the robot's velocity ($v/v_n/\omega$) relative to its reference (v_r, $v_{n,r}$, and ω_r) also resembles a first-order system, given that the first-order approximation of the motors' angular speed is used in a cascade control scheme. The integration of $v/v_n/\omega$ estimates the robot's pose on a coordinate frame aligned with the local frame but with the same origin

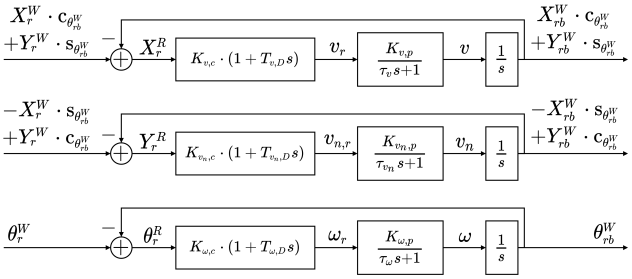


Fig. 7 PD controllers for the robot's pose considering a first-order approximation of systems $v/v_n/\omega = f(v/v_n/\omega_r)$

as the world frame ($\{X^W, Y^W\}$). Matrix $R^{-1}(\theta^W_{rb})$ (see equation 1) describes the homogeneous transformation between these two coordinate frames (only a pure rotation due to the frames having the same origin). Figure 7 presents the block diagram for the PD controllers to control the robot's velocity. The error of these controllers is the trajectory pose reference for the robot's on its local coordinate frame ($\{X^R, Y^R\}$). Also, note that each control variable v_r, $v_{n,r}$, and ω_r can be controlled independently due to the holonomic characteristics of omnidirectional robots [18].

The parameters of the PD controllers are the proportional gain ($K_{j,c}$) and the derivative time ($T_{j,D}$ where $j = v, v_n, \omega$), where these parameters depend on the desired closed-loop poles. These poles can be chosen considering the roots of normalized Bessel polynomials corresponding to a settling time ($T_{j,sett.}$) of 1 second. Given the consideration of first-order systems for the robot's velocity and that the position control is a second-order system, the poles defined by Bessel polynomials are $p = -4.0530 + j2.3400$ and $p^* = -4.0530 - j2.3400$. Equation 6 defines the closed-loop characteristic polynomial of a second-order system for the pole p and its conjugate p^*, where $T_{j,sett.}$ should be the lowest value possible considering the limits of the discrete real system (e.g., the control period). Equation 7 defines the characteristic polynomial for the PD controllers. Finally, equation 8 defines the parameters of the PD controllers.

$$s^2 - \frac{2 \cdot \mathrm{Re}\{p\}}{T_{j,sett.}} \cdot s + \frac{|p|^2}{{T_{j,sett.}}^2} = 0 \tag{6}$$

$$s^2 + \frac{K_{j,c}K_{j,p}T_{j,D} + 1}{\tau_j} \cdot s + \frac{K_{j,c}K_{j,p}}{\tau_j} = 0 \tag{7}$$

$$\begin{cases} K_{j,c} = \frac{\tau_j}{K_{j,p}} \cdot \frac{|p|^2}{{T_{j,sett.}}^2} \\ T_{j,D} = \frac{-2 \cdot \mathrm{Re}\{p\} \cdot \tau_j / T_{j,sett.} - 1}{K_{j,c}K_{j,p}} \end{cases} \tag{8}$$

In order to set the PD controllers' parameters, it was performed experiments to retrieve the first-order parameters of

Table 3 Statistic measures of the 1st-order approximation (estimate parameters $K_{j,p}$ and τ_j, where $j = v, v_n, \omega$) for the responses of v, v_n, and ω to a step change in their references with initial velocities different than 0 (μ: mean; σ: standard deviation; m: median; min: minimum; max: maximum)

	$K_{j,p}$			τ_j (s)		
	v	v_n	ω	v	v_n	ω
μ	1.0231	1.0251	0.9804	0.1148	0.1044	0.0873
σ	0.0215	0.0184	0.0118	0.0303	0.0317	0.0180
m	1.0196	1.0314	0.9787	0.1212	0.1057	0.0908
min	0.9917	0.9821	0.9632	0.0341	0.0447	0.0411
max	1.0625	1.0435	1.0055	0.1646	0.1448	0.1096

the responses of v, v_n, and ω to a step change in their references. The robot had initial velocities different than 0 and, for each type of independent motion v, v_n, and ω, it was put a step change in the respective velocity reference (10 experiments for each type of motion). Similar to the analysis for the angular speed control models, an optimization algorithm was used to approximate the response to a first-order system and retrieve the systems' parameters ($K_{j,p}$ and τ_j, where $j = v, v_n, \omega$). Table 3 presents statistical measures of the estimated parameters for each type of motion v, v_n, and ω. The time constants of the robot's velocity were defined as the median of the obtained results to provide a robust estimation of these constants. As for the gains of the first-order approximation, their values were set as 1.0000. Even though both median and mean estimators give a value different than 1.0000, these parameters define the gain between the reference and the actual value of v, v_n, and ω which should be 1. Then, it was considered a settling time ($T_{j,sett.}$) of 0.7s for the three PD controllers (figure 7) and their parameters were computed using equation 8:

$$\begin{bmatrix} K_{v,c} & K_{v_n,c} & K_{\omega,c} \end{bmatrix} = \begin{bmatrix} 5.4176 & 4.7225 & 4.0598 \end{bmatrix}$$
$$\begin{bmatrix} T_{v,D} & T_{v_n,D} & T_{\omega,D} \end{bmatrix} = \begin{bmatrix} 0.0745\text{s} & 0.0473\text{s} & 0.0128\text{s} \end{bmatrix}$$

One important observation is that the median values of the time constants for each motion are lower than the closed-loop time constant (i.e., the time constant when the PI controllers are enabled for angular speed control) that we set for the PI controllers (0.1212s, 0.1057s, and 0.0908s for v, v_n, and ω, respectively, when compared to $\tau_{\dot{\varphi},cl} = 0.1392$s). Furthermore, considering as reference the value used for the PI controllers ($\tau_{\dot{\varphi},cl} = \tau_{\dot{\varphi}}/1.5$), we have the following ratios between the open-loop mean value and the closed-loop median values of the time constants: 2.12 (0.2564s/0.1212s), 2.36 (0.2497s/0.1057s), and 1.32 for v, v_n, and ω, respectively. Even though the one for ω is lower than the one set for computing the PI parameters (1.32 ¡ 1.5), note that it was averaged the estimated parameters of the angular speed's model (i.e., different motions were not treated differently). These results demonstrate that our approach led to time constants equal or lower than the one set for closed-loop independently of the type of motion.

However, as analyzed in section 5, the pose control with only PD controllers lead to a delay between the reference and the actual value of the controlled variables.

4 Pose Control with Feed-Forward (FF) Controllers

Therefore, we propose the use of FF controllers to reduce the delay between the reference and the actual value of the robot pose. FF controllers do not affect the stability of the system. In addition, given that these controllers require the derivatives of the reference (first and second derivatives corresponding to the robot's desired velocity and acceleration, respectively), FF controllers have predictive characteristics. The final control scheme for omnidirectional robots proposed in this article is illustrated in figure 8.

4.1 Feed-forward parameters

FF controllers invert the transfer function of a system assuming that we know the system's model. Given the first-order nature of the models analyzed in this article, equation 9 defines the transfer function of FF controllers for first-order systems. These controllers were implemented in the proposed control scheme as shown in figure 8. Note that the output of each FF controller adds to the outputs of the PD controllers to compute the references for the linear (v_r and $v_{n,r}$) and angular (ω_r) velocities of the robot. The parameters of the robot's velocity model ($K_{j,p}$ and τ_j, where $j = v, v_n, \omega$) are already estimated in Section 3.

$$H_j(s) = \frac{\tau_j}{K_{j,p}} \cdot s^2 + \frac{1}{K_{j,p}} \cdot s \tag{9}$$

4.2 Computation of the derivatives

Given that our trajectory T is a set of points ($T = \{P_{r,0}(t_0), P_{r,1}(t_1), \ldots, \{P_{r,k}(t_k), \ldots, P_{r,N-1}(t_{N-1})\}$), we already know the future poses desired for the robot at a time instant t_k. Consequently, we can approximate a subset of points of the trajectory into parametric polynomials to estimate the derivatives required for the FF controllers at that time instant. Furthermore, the time interval between consecutive points is considered to be constant (see Section 2). So, we can normalize this interval and define it as 1 on a time domain u. The transformation $t \rightarrow u$ is characterized by equation 10, where t_l represents the time instant of the trajectory's current reference pose desired for the robot and K_T is the transformation scalar between the two time domains. The constant

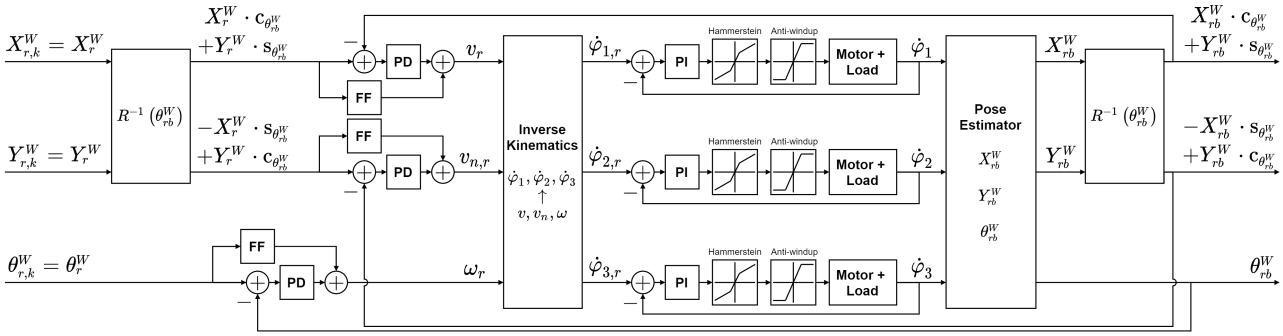


Fig. 8 Proposed trajectory controller for omnidirectional robots

K_T normalizes the time interval between poses (e.g., when performing 2 consecutive trajectories with different time intervals between consecutive poses).

$$u = K_T \cdot (t - t_l) \tag{10}$$

In order to compute the derivatives of the FF controllers, first, we define the future trajectory as a subset of points F (where $F = \{P_{r,l+0}(t_{l+0}), \ldots, P_{r,l+M-1}(t_{l+M-1})\}$ and $F \subset T$) with M elements on the time domain u (see equation 10). These elements represent the future poses relative to the current one of the robot on the world coordinate frame. Next, the second derivative ($\ddot{f}_h$ where $h = X^W_r, Y^W_r, \theta^W_r$) equivalent to the reference's acceleration is computed using a second-degree polynomial approximation of the future trajectory, as illustrated in equation 11. The approximation is restricted by the initial position ($a_{h,0}$) and velocity ($a_{h,1}$) estimated based on the future trajectory, as defined in equation 12. Finally, equation 13 defines the least-squares solution with the Monroe-Penrose inverse matrix ($[\ldots]^\dagger$) for the coefficients $a_{h,2}$.

$$\begin{aligned} f_h(u) &= a_{h,0} + a_{h,1} \cdot u + (1/2) \cdot a_{h,2} \cdot u^2 \\ \dot{f}_h(u) &= a_{h,1} + \cdot a_{h,2} \cdot u \\ \ddot{f}_h(u) &= a_{h,2} \end{aligned} \tag{11}$$

$$\begin{cases} f_h(0) = a_{h,0} \\ \dot{f}_h(0) = a_{h,1} \end{cases} \Leftrightarrow \begin{cases} a_{h,0} = P_{h,l+0} \\ a_{h,1} \approx P_{h,l+1} - P_{h,l+0} \end{cases} \tag{12}$$

$$\begin{aligned} &\begin{bmatrix} P_{h,l+0} - a_{h,0} - a_{h,1} \cdot 0 \\ P_{h,l+1} - a_{h,0} - a_{h,1} \cdot 1 \\ \vdots \\ P_{h,l+M-1} - a_{h,0} - a_{h,1} \cdot (M-1) \end{bmatrix} = \begin{bmatrix} 0^2/2 \\ 1^2/2 \\ \vdots \\ (M-1)^2/2 \end{bmatrix} \cdot a_{h,2} \\ \Leftrightarrow\ & a_{h,2} = \begin{bmatrix} 0^2/2 \\ 1^2/2 \\ \vdots \\ (M-1)^2/2 \end{bmatrix}^\dagger \cdot \begin{bmatrix} P_{h,l+0} - a_{h,0} - a_{h,1} \cdot 0 \\ P_{h,l+1} - a_{h,0} - a_{h,1} \cdot 1 \\ \vdots \\ P_{h,l+M-1} - a_{h,0} - a_{h,1} \cdot (M-1) \end{bmatrix} \end{aligned} \tag{13}$$

The first derivative ($\dot{g}_h$ where $h = X^W_r, Y^W_r, \theta^W_r$) is estimated using a first-order approximation polynomial illustrated in equation 14. The coefficients of the first-order approximation ($b_{h,0}$ and $b_{h,1}$) are estimated using the least-squares algorithm where $b_{h,1}$ represents the velocity of the robot pose's reference, as illustrated in equation 15.

$$\begin{aligned} g_h(u) &= b_{h,0} + b_{h,1} \cdot u \\ \dot{g}_h(u) &= b_{h,1} \end{aligned} \tag{14}$$

$$\begin{aligned} &\begin{bmatrix} P_{h,l+0} \\ P_{h,l+1} \\ \vdots \\ P_{h,l+M-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ \vdots & \vdots \\ 1 & (M-1) \end{bmatrix} \cdot \begin{bmatrix} b_{h,0} \\ b_{h,1} \end{bmatrix} \\ \Leftrightarrow\ & \begin{bmatrix} b_{h,0} \\ b_{h,1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ \vdots & \vdots \\ 1 & (M-1) \end{bmatrix}^\dagger \cdot \begin{bmatrix} P_{h,l+0} \\ P_{h,l+1} \\ \vdots \\ P_{h,l+M-1} \end{bmatrix} \end{aligned} \tag{15}$$

An important note for both first and second-order approximations is that the orientation of the points in F must be unwrapped to compute the polynomial approximations. When the orientation is unwrapped, the orientation data does not have discontinuities. This operation is required for avoiding wrong parameter estimations by the linear least-squares algorithm. Also, note that the Monroe-Penrose inverse matrices required for computing both first and second-order derivatives approximation are constant. Indeed, these matrices can be computed offline (e.g., when the robot's control module starts up) and are not require to be updated.

In figure 9, it is possible to visualize the difference between approximating a future trajectory ($M = 7$) on a square corner by a second-order polynomial with (w/) and without (wo/) the restrictions of initial position and velocity. The first observation is that estimating the velocity and the acceleration from a second-order approximation without any restrictions would not predict correctly (at least, at an initial stage) the future behavior of the trajectory. Second, the acceleration ($\{a^W_{X,2}, a^W_{Y,2}\}_{2nd-order}$) and velocity ($\{b^W_{X,1}, b^W_{Y,1}\}_{1st-order}$) vectors estimated with the proposed approach predicts correctly the behavior of the reference: on the robot's arrival

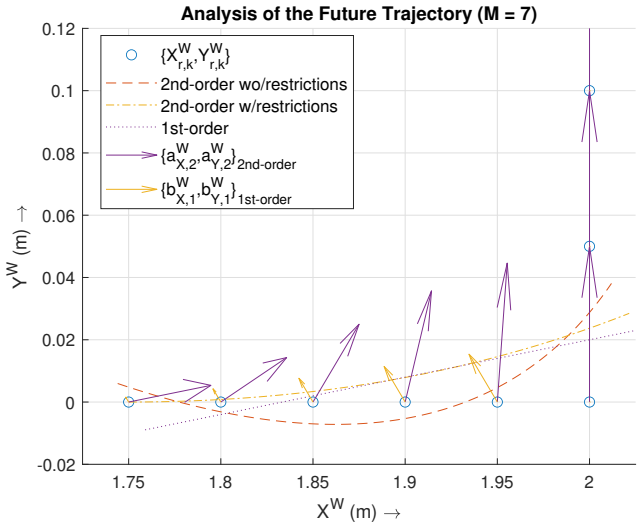
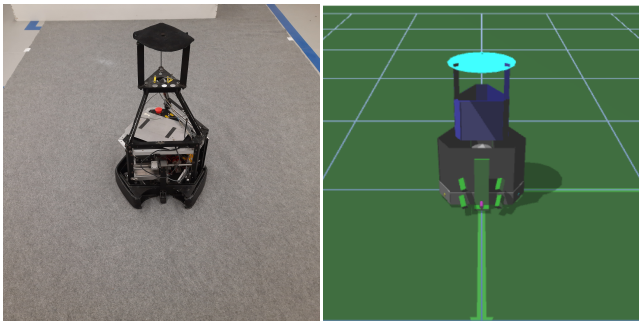


Fig. 9 Analysis of the future trajectory in XY on a square corner

to the square corner, the velocity starts to increase perpendicular to the arrival direction, as intended; on the exit of the corner, the derivatives estimate a zero acceleration due to the points being equidistant and the time interval between consecutive points a constant one, just as expected. Note that higher orders approximations would lead to oscillations on the approximated curves, and that would cause unwanted oscillations on the pose control. Also, the analysis of the derivatives effect for the robot's orientation would be similar to the one presented in figure 9, but only on one dimension.

Another important remark is when the last element of F is the same one as the last desired pose of the trajectory T, i.e., $F = \{P_{r,N-M}(t_{N-M}), \ldots, P_{r,N-M+l}(t_{N-M+l}), \ldots, P_{r,N-1}(t_{N-1})\}$. From this instant onward (where $k \geq N-M$, until the robot reaches the desired goal, i.e., $P_{r,N-1}$), we defined that the poses of F with index $N-k-1 \leq l \leq M-1$ should be equal to $P_{r,N-1}$. This approach means that the robot is arriving to a constant final pose for $k \geq N-M$ leading to similar behavior as the one presented in figure 9 for the X^W coordinate: its value remains constantly equal to 2m (after the square corner) originating a negative deceleration and consequent decrease in velocity (in the X^W direction) when arriving to the corner, just as we intend when the robot's is arriving to its goal.

Lastly, the first and second derivatives should be multiplied by K_T and K_T^2, respectively (due to the feed-forward derivatives being defined in the time domain of t). Also, the coefficients $a_{h,2}^W$ and $b_{h,1}^W$ are relative to the world coordinate frame. So, these coefficients must be multiplied by the matrix $R^{-1}(\theta_{rb}^W)$ (equation 1) to be used in the FF controllers formulated in this section (and also illustrated in figure 8).



(a) (b)

Fig. 10 Three-wheeled omnidirectional robot of the 5DPO Middle Size League robot soccer [14]: (a) real robot; (b) simulation environment (SimTwo [5] – ODE-based simulator)

5 Experiments

The control scheme proposed in Section 4 was implemented for the three-wheeled omnidirectional robot of the 5DPO Middle Size League robot soccer [14]. This robot is illustrated in figure 10 and has as kinematic parameters l and D of 0.191367m and 0.098566m, respectively. First, a SimTwo [5] simulation environment (uses the ODE physics engine) with a simulated version of this robot was adjusted based on the angular speed responses of the motors analyzed in Section 3. The simulator was used to study the size of the future trajectory for three different trajectories – a square and an 8 with constant orientation, and a square with variable orientation along the trajectory – and compare the time responses of the robot's pose using the proposed controlling scheme versus the use of only PD controllers for pose control. Then, the control scheme was implemented in a ROS [16] node that subscribed to the odometry topic of the robot (the robot did not have available other pose estimators) and published the robot's linear and angular velocities through the topic `cmd_vel`. The computer used was an Asus X571GT laptop with an Intel Core i7-9750H 6c/12t 2.60–4.50GHz, 12GB of DDR4 RAM, and an NVIDIA GTX 1650M 4GB of GDDR5 VRAM video memory. The experiments with the real robot focused on validating the study of the future trajectory's size performed in SimTwo [5] and evaluating the usage of the pose control period (in the time domain) for different sizes of the trajectory. All experiments had the angular motor speed control at 100Hz and the pose control at 25Hz.

5.1 Simulation results

5.1.1 Size of the future trajectory

The first experiment was to study the size of the future trajectory in the SimTwo [5] simulation environment. The trajectories used in the experiments complied with the require-

ment discussed in Section 2 (with initial acceleration and deceleration of 1.0m.s^{-2}, a certain nominal velocity, and a final nominal velocity of approximation to the desired goal of 0.10m.s^{-1}) and were the following ones:

- 2x2m square with a constant 0º orientation;
- 2x2m square with variable orientation (the same one illustrated in figures 2 and 3);
- 8-shape with a radius of 1m and a constant 0º orientation.

The robot followed the 3 trajectories with sizes of the future trajectory (M) from 3 to 22 points. The distance between consecutive points of the generated trajectories was 0.04m at nominal velocity, and this velocity is defined by the time interval between two consecutive points.

Figure 11 presents the evaluation of the controller's performance for different values of M (3..22) at different nominal velocities (0.5, 0.75, and 1.0m.s^{-1}) over the 3 trajectories. The evaluation was based on the following error heuristic (ε_h):

$$\varepsilon_h = \frac{\varepsilon_{d,avg} + \varepsilon_{d,avg} + \varepsilon_{t,max} + \varepsilon_{t,max}}{T_d} + \frac{\varepsilon_{\theta,avg} + \varepsilon_{\theta,max}}{T_\theta} \quad (16)$$

where $\varepsilon_{e,avg}$ and $\varepsilon_{e,max}$ represent the average and maximum values of an error quality measure, respectively, and $\varepsilon_{d,avg/max}$ (m) is the distance error over time, $\varepsilon_{\theta,avg/max}$ (º) is the absolute orientation error over time, and $\varepsilon_{t,avg/max}$ (m) is the trajectory error (distance between the robot's position and the closest point of the desired trajectory T). The tolerances for following the trajectory in terms of distance (T_d, in meters) and orientation (T_θ, in degrees) were defined as 0.025m and 2.5º, respectively. These tolerances should be defined based on the use application for the robots (in our case, soccer mobile robots). Also, note that T_d and T_θ normalize the distance and orientation error quality measures required for the error heuristic (ε_h) presented in equation 16.

Analyzing figure 11, the first observation is that the future size M influences the error quality measures on pose control independently of the trajectory considered. For lower values of M, the robot cannot foresee the changes in the trajectory leading to the deviation of the robot from the desired trajectory (e.g., having an overshoot on each corner in a squared trajectory). For higher values of M, the robot starts to filter the sudden changes in the path deviating from the desired trajectory (e.g., cut corners). Another observation is the influence of the desired nominal velocity over the path. This influence is illustrated in the figure by the shift of the curve $\varepsilon_h(M)$ to the right (also higher error values, but it is expected due to the inertia of the robot itself) meaning that higher velocities require greater predictability of the trajectory's changes from the computation of the FF derivatives.

Furthermore, the curves $\varepsilon_h(M)$ show optimal values for M depending on the robot's nominal velocity. The following optimal values of M were obtained for each of the 3 trajectories tested in simulation: 5, 8, and 11 in the 2x2m square with constant 0º orientation (sq), 5, 8, and 10 in the 2x2m square with variable orientation ($\text{sq}_{\text{w/}\theta^{w}\text{var.}}$), and 5, 7, and 9 in the 8-shape with a 1m radius (8) for the nominal velocities of 0.50, 0.75, and 1.00m.s^{-1}, respectively. However, the optimal value should be analyzed in distance (M_d) and time (considering the nominal velocity), as follows:

- $M = 5 \rightarrow M_d = 0.20\text{m} \xrightarrow{0.50\text{m/s}} \Delta t = 0.40s$
- $M = 7 \rightarrow M_d = 0.28\text{m} \xrightarrow{0.75\text{m/s}} \Delta t = 0.37s$
- $M = 8 \rightarrow M_d = 0.32\text{m} \xrightarrow{0.75\text{m/s}} \Delta t = 0.43s$
- $M = 9 \rightarrow M_d = 0.36\text{m} \xrightarrow{1.00\text{m/s}} \Delta t = 0.36s$
- $M = 10 \rightarrow M_d = 0.40\text{m} \xrightarrow{1.00\text{m/s}} \Delta t = 0.40s$
- $M = 11 \rightarrow M_d = 0.44\text{m} \xrightarrow{1.00\text{m/s}} \Delta t = 0.44s$

Table 4 presents the error quality measures for the 3 trajectories and the 3 nominal velocities using the optimal values of M. All maximum error measures on the square trajectories were higher than the tolerances considered for the error heuristic ε_h (0.025m and 2.5º for distance and orientation error measures, respectively). However, the average values had better results: only at 1.00m/s $\varepsilon_{d,avg}$ and $\varepsilon_{\theta,avg}$ were greater than the tolerances (0.031m and 3.30º for the squared trajectory with constant orientation, and 0.036m and 4.52º for the other squared trajectory). Even though the highest maximum absolute orientation error on the squared trajectories was 20.8º, it should be noted that a squared trajectory does not have continuous reference velocity even if its module is continuous. The discontinuity occurs in the corners where the direction of the robot's velocity has a step change. Considering those discontinuities and the high nominal velocities used in the experiments (especially in the case of 1.00m.s^{-1}), we consider that our controller was able to follow the squared trajectories acceptably with reasonable results. As for the 8-shaped trajectory, all the distance and orientation errors were lower than the trajectory tolerances considered for formulating ε_h. Even at 1.00m.s^{-1}, the average distance ($\varepsilon_{d,avg}$) and trajectory ($\varepsilon_{t,avg}$) errors were 0.014m and 0.012m, respectively, and the absolute orientation average error ($\varepsilon_{\theta,avg}$) was 0.51º. These results were at least $\sim$50% lower than the tolerances. In terms of maximum errors on the 8-shaped trajectory, the tolerances were only exceeded at 1.00m/s for the maximum distance and trajectory errors (higher 0.006m and 0.005m, respectively).

In terms of setting an optimal value for M, the simulation results for all 3 trajectories at a nominal velocity of 0.50m.s^{-1} demonstrate that M should be 5 (a time window of 0.40s). Next, the squared trajectories resulted in the optimal value of $M = 8$ and the 8-shape in $M = 7$ at 0.75m/s. Given the 0.62% difference in the error heuristic (ε_h) with $M = 8$ versus 7 on the 8-shaped trajectory (2.422 and 2.407, respectively), we consider $M = 8$ (equivalent to a 0.43s time

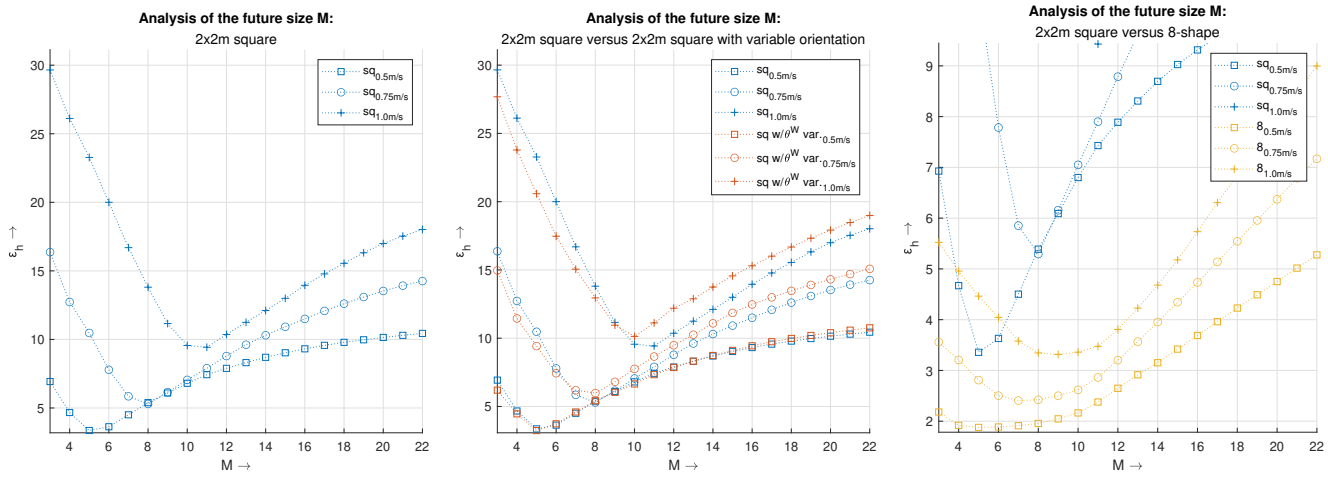


Fig. 11 Analysis of the error heuristic (ε_h) on 3 different trajectories (2x2m square with a constant 0º orientation, 2x2m square with variable orientation – the same one illustrated in figures 2 and 3, and an 8-shape with a radius of 1m and a constant 0º orientation) at different nominal velocities (0.5, 0.75, and 1.0m.s^{-1}) depending on the size of the future trajectory (M, 3..22)

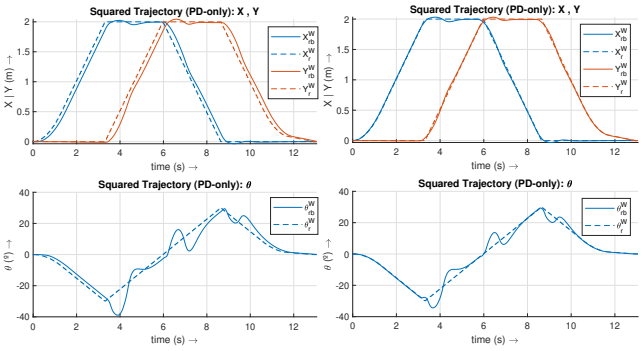
Table 4 Error quality measures ($\varepsilon_{d,avg/max}$ (m) – distance error over time, $\varepsilon_{\theta,avg/max}$ – absolute orientation error over time, $\varepsilon_{t,avg/max}$ (m) – trajectory error, and ε_h – error heuristic) for the simulation experiments with optimal values of M over the 3 trajectories tested

			Average			**Maximum**			
	v		$\varepsilon_{d,avg}$	$\varepsilon_{\theta,avg}$	$\varepsilon_{t,avg}$	$\varepsilon_{d,max}$	$\varepsilon_{\theta,max}$	$\varepsilon_{t,max}$	
Trajectory	(m/s)	M	(m)	(º)	(m)	(m)	(º)	(m)	ε_h
sq	0.50	5	0.007	0.25	0.005	0.044	5.05	0.027	3.357
	0.75	8	0.014	0.95	0.009	0.062	9.24	0.046	5.293
	1.00	10	0.031	3.30	0.018	0.116	18.6	0.071	9.558
	1.00	11	0.030	3.12	0.018	0.109	19.6	0.075	9.429
sq$_{w/\theta^W var.}$	0.50	5	0.008	0.42	0.006	0.041	4.79	0.025	3.233
	0.75	8	0.017	1.59	0.012	0.070	9.12	0.048	5.981
	1.00	10	0.034	4.26	0.022	0.122	18.3	0.071	10.13
	1.00	11	0.036	4.52	0.024	0.131	20.8	0.083	11.12
8	0.50	5	0.006	0.22	0.005	0.020	0.76	0.015	1.877
	0.75	7	0.009	0.35	0.008	0.024	0.84	0.020	2.407
	0.75	8	0.009	0.35	0.007	0.024	0.84	0.020	2.422
	1.00	9	0.012	0.51	0.011	0.031	0.94	0.030	3.316
	1.00	10	0.012	0.51	0.011	0.031	0.92	0.030	3.360
	1.00	11	0.014	0.51	0.012	0.031	0.91	0.030	3.475

window) as its optimal value when the robot is moving at a nominal velocity of 0.7m.s^{-1}. As for the nominal velocity of 1.00m.s^{-1}, three different optimal values of M were obtained: 11 for the squared trajectory with constant orientation, 10 for the other squared trajectory, and 9 for the 8-shaped one. Given the time windows obtained at 0.50m.s^{-1} and 0.7m.s^{-1}, the expected time window should be around 0.40–0.43s; consequently, $M = 10$ and $M = 11$ at 1.00m/s (representing time windows of 0.40s and 0.44s, respectively). The differences in terms of ε_h between these two sizes are 1.35%, 9.77%, and 3.42% on the squared trajectory, the other squared trajectory, and the 8-shaped one, respectively. Considering a lower ε_h with $M = 10$ versus 11 on the squared trajectory with variable orientation and considering the differences between these two sizes on the other two trajectories, we considered $M = 10$ as the optimal value for the future's trajectory size at 1.00m.s^{-1}. Therefore, M should be equal to the value that includes a subset of points of the trajectory equivalent to approximately 0.40s ($M = 5, 8, 10$ for the nominal velocities of 0.50m.s^{-1}, 0.75m.s^{-1}, and 1.00m.s^{-1}, respectively). Still, this time window of 0.40s should be validated in real experiments.

5.1.2 Comparison with PD-only position control

Next, the comparison of the proposed controller to only using PD controllers for the robot's pose on the squared trajectory with variable orientation is illustrated in figure 12. The nominal velocity was 0.75m^{-1} and the value of M (when using the proposed controller) was 8. When using only PD controllers, the actual pose of the robot is delayed relative to the desired one. Indeed, the delay of approximately 0.20s for $\{X^W, Y^W\}$ and 0.24s for θ^W leads to a $\varepsilon_{d,max}$ and a $\varepsilon_{\theta,max}$ of 0.238m and 16.1º, respectively (compared to 0.070m and 9.12º for the proposed controller). However, $\varepsilon_{t,max}$ with PD-only pose control is still similar to the proposed control scheme: 0.050m versus 0.048m, respectively. This similarity is expected because $\varepsilon_{t,avg/max}$ only evaluates the error in space.



(a) (b)

Fig. 12 Comparison of the proposed controller with a PD-only control: (a) PD-only; (b) proposed controller

5.2 Real experiments

5.2.1 Validation of the future trajectory's size

The first goal of the experiments with the 5DPO [14] three-wheeled omnidirectional robot (see figure 10) was to validate the study of the future trajectory's size M performed in the SimTwo [5] simulation environment. The trajectories used were the same ones as in the simulation. First, the squared trajectory with constant orientation was evaluated for different values of M and the same nominal velocities used in SimTwo [5] (0.50, 0.75, and 1.00m.s^{-1}). Then, the square with variable orientation and the 8-shape trajectories were tested at 0.75 and/or 1.00m.s^{-1} for different values of M. Figure 13 presents the evaluation of the controller's performance for the three trajectories based on the curves $\varepsilon_h(M)$ versus the ones obtained in simulation.

Although the shape of the curves $\varepsilon_h(M)$ is similar to the ones obtained in the simulated experiments (see figure 11), the optimal values for M are different for the same nominal velocities. Indeed, the optimal values obtained for M with the real robot were the following ones: 8 (0.32m), 12 (0.48m), and 13/15 (0.52/0.60m) for the nominal velocities of 0.50, 0.75, and 1.00m.s^{-1}, respectively. Analyzing these optimal values in distance and in time, we obtained the following results:

- $M = 8 \rightarrow M_d = 0.32\text{m} \xrightarrow{0.50\text{m/s}} \Delta t = 0.64s$
- $M = 12 \rightarrow M_d = 0.48\text{m} \xrightarrow{0.75\text{m/s}} \Delta t = 0.64s$
- $M = 13 \rightarrow M_d = 0.52\text{m} \xrightarrow{1.00\text{m/s}} \Delta t = 0.52s$
- $M = 15 \rightarrow M_d = 0.60\text{m} \xrightarrow{1.00\text{m/s}} \Delta t = 0.60s$

The maximum time interval obtained was 0.64s being 0.24s higher than the one obtained in simulation (0.40s). This difference is probably due to unmodelled effects of the simulated robot. Also, note that we obtained two different optimal values for M in the case of a nominal velocity of 1.00m.s^{-1}: 13 (squared trajectories) and 15 (8-shape). At this nominal velocity, the time windows for $M = 13$ (0.52s) and 15 (0.60s) are lower than at 0.50 or 0.75m.s^{-1} (0.64s for both cases), i.e., lower 0.12s and 0.04s, respectively. Indeed, the value that we were expecting to obtain for M at 1m.s^{-1} would be 16 (0.64s $\xrightarrow{1.00\text{m/s}} M_d = 0.64\text{m} \rightarrow M = 16$). However, the differences between having M equal to 13 or 16 for the squared trajectory and 15 or 16 for the 8-shape one are 14.2% and 3.76%, respectively, in terms of the error heuristic ε_h (8.871 vs 10.13 and 5.320 vs 5.520, respectively). In the case of the 14.2% difference, ε_h for $M = 16$ is greater than for $M = 11$ due to the trajectory tolerances considered for the error heuristic (0.025m and 2.5º for distance and orientation error measures, respectively). Given the difference between the error metrics for these two cases being lower than 0.014m and 0.01º, we acknowledge that it is very similar using $M = 11$ or $M = 16$ for the squared trajectory at 1.00m.s^{-1}. So, we consider that the optimal value for the future trajectory's size M should be the one that corresponds to a time window of 0.64s. In our case (the robot that we were using and the 3 trajectories considered in this article), $M = 8, 12, 16$ for nominal velocities of 0.50, 0.75, and 1.00m.s^{-1}, respectively. Note that for other velocities M must change its value to consider the 0.64s time window.

As for analyzing the error quality measures, table 5 presents the obtained results in the real experiments for the following values of M: the optimal values 5, 8, and 10 (0.50, 0.75, and 1.00m.s^{-1}) defined based on the simulation results; the optimal values 8, 12, and 13/15 (0.50, 0.75, and 1.00m.s^{-1}) obtained in the real experiments; and the expected value of 16 for the nominal velocity of 1.00m.s^{-1}. As already discussed, the optimal values of M for the real robot differed from the simulation results.

For squared trajectories, the maximum differences of the error metrics were 0.04m (at 0.50m.s^{-1}, $\varepsilon_{d,max} = 0.104$m and 0.064m for $M = 5$ and 8) and 1.06º (at 0.75m.s^{-1} on the squared trajectory with variable orientation, $\varepsilon_{\theta,max} = 4.13$º and 3.07º for $M = 10$ and 13) for distance and orientation measures, respectively. For the 8-shape trajectory, the maximum differences were 0.018m (at 1.00m.s^{-1}, $\varepsilon_{d,max} = 0.058$m for $M = 15$ vs 0.076m and 0.074m for $M = 10$ and 13) and 1.06º (at 1.00m.s^{-1}, $\varepsilon_{\theta,max} = 2.06$º for $M = 15$ vs 3.12º and 2.54º for $M = 10$ and 13). These results show that the controller was robust for different sizes of the future trajectory considering the tolerances considered for the trajectory tracking errors. In terms of using $M = 16$ at 1.00m.s^{-1} compared to $M = 13$, the maximum differences of the error measures were 0.014m ($\varepsilon_{d,max} = 0.121$m vs 0.107m on the squared trajectory) and 0.54º ($\varepsilon_{\theta,max}$ =2º vs 2.54º on the 8-shape trajectory) for distance and orientation metrics, respectively. Also, analyzing the average error measures, only the error metric $\varepsilon_{d,avg}$ at 1.00m.s^{-1} on the squared trajectory was greater than the tolerance of 0.025m. Thus, the usage of $M = 16$ for velocities of 1.00m.s^{-1} is valid generalizing the

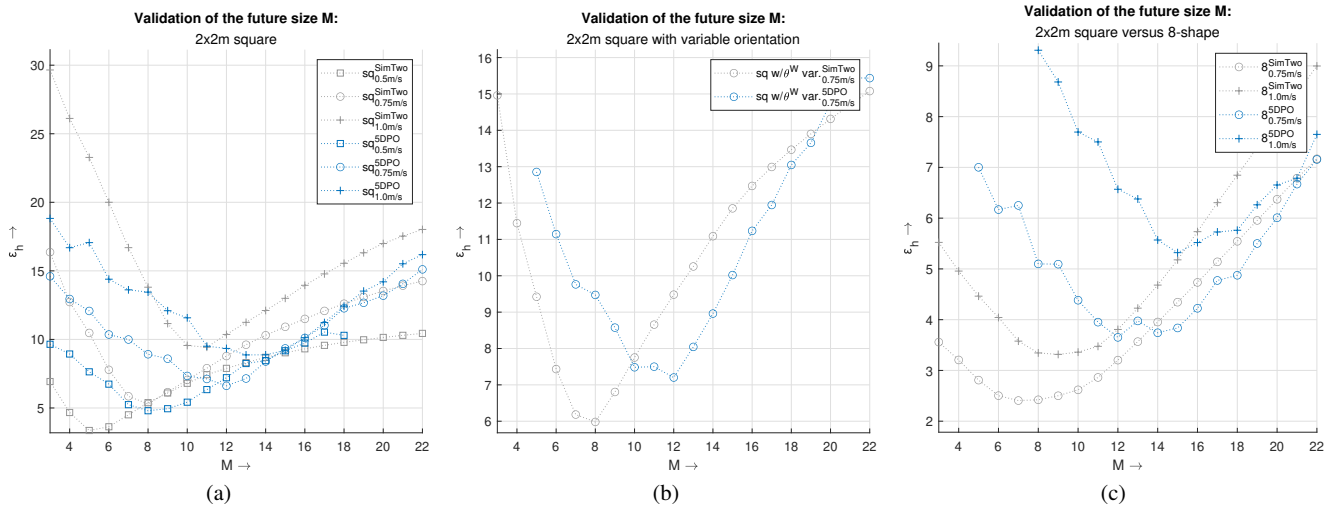


Fig. 13 Validation of the analysis of the error heuristic (ε_h) presented in figure 11 for the 5DPO [14] omnidirectional robot (blue curves) versus the simulation results (grey curves)

Table 5 Error quality measures ($\varepsilon_{d,avg/max}$ (m) – distance error over time, $\varepsilon_{\theta,avg/max}$ – absolute orientation error over time, $\varepsilon_{t,avg/max}$ (m) – trajectory error, and ε_h – error heuristic) for the validation experiments of the optimal values of M using the real robot

			Average			**Maximum**			
	v		$\varepsilon_{d,avg}$	$\varepsilon_{\theta,avg}$	$\varepsilon_{t,avg}$	$\varepsilon_{d,max}$	$\varepsilon_{\theta,max}$	$\varepsilon_{t,max}$	
Trajectory	(m/s)	M	(m)	(º)	(m)	(m)	(º)	(m)	ε_h
sq	0.50	5	0.015	0.27	0.008	0.104	1.55	0.064	7.637
	0.50	8	0.012	0.32	0.006	0.064	1.64	0.038	4.799
	0.75	8	0.023	0.48	0.012	0.120	1.61	0.068	8.917
	0.75	12	0.020	0.41	0.011	0.082	1.85	0.052	6.616
	1.00	10	0.034	0.59	0.019	0.145	2.36	0.092	11.58
	1.00	13	0.030	0.56	0.018	0.107	1.85	0.067	8.871
	1.00	16	0.032	0.56	0.020	0.121	1.84	0.080	10.13
$\text{sq}_{\text{w/}\theta^W\text{ var.}}$	0.75	8	0.026	0.68	0.016	0.122	4.13	0.071	9.47
	0.75	12	0.022	0.61	0.014	0.085	3.07	0.058	7.202
8	0.75	8	0.024	0.45	0.017	0.055	1.85	0.031	5.099
	0.75	12	0.017	0.41	0.009	0.043	1.61	0.022	3.652
	1.00	10	0.039	0.68	0.027	0.076	3.12	0.050	7.696
	1.00	13	0.030	0.71	0.019	0.074	2.54	0.036	6.378
	1.00	15	0.026	0.60	0.014	0.058	2.06	0.034	5.320
	1.00	16	0.024	0.63	0.012	0.064	2.00	0.037	5.520

definition of M for the 5DPO [14] three-wheeled omnidirectional robot based on the 0.64s reference time window.

5.2.2 Processing time usage of the controller

Lastly, the processing time usage of the controller was analyzed using the `high_resolution_clock` class[1] from the chrono C++ library. Figure 14 presents the distribution of the processing time usage for different values of M and in stand-by (when the controller is not following a certain trajectory, the derivatives are not required to be computed) with at least 1000 samples for each experiment. First, the high number of outliers present in the graphic are not dependent on the controller but on the operating system and the hardware itself. Even though the ROS [16] node implemented used only 1 thread, the task scheduling of the operating system would vary the executing time for each cycle of the controller. Second, the execution time seems to have a linear relation with M. Using the median as reference, the processing time data (Δt_{usage}, in milliseconds) fits a regression line with a R-squired of 0.9964 defined as in equation 17. Even though the obtain equation is hardware-dependent, it can be used as reference a median estimation of the processing time usage for values M not analyzed in this article (if the computer used has the same specifications as the one used to implement the ROS [16] node in the real experiments). Finally, the percentage of the maximum processing time usage was 1.41% ($\Delta t_{\text{usage}} = 0.562291$ms for $M = 24$) considering a pose control period of 40ms (25Hz). This result shows that the proposed control scheme is ideal for low-cost applications in terms of computation resources or systems with fast dynamics. Indeed, the pose control period could be increased up to 100Hz (the same frequency that the PI controllers were executed in the real and simulated experiments) and the time usage percentage would be lower than 56.2%. However, the hardware used in the experiments was limited by the serial port's maximum data transmission and it was not possible to test this situation. Also, note that these results benefited from the offline computation of the matrices of the linear least-squares solutions required to approximate the future trajectory.

[1] https://www.cplusplus.com/reference/chrono/high_resolution_clock/

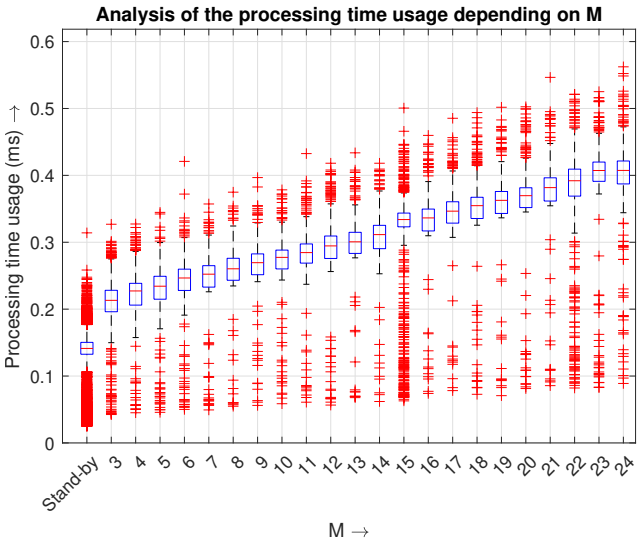


Fig. 14 Analysis of the processing time usage depending on the size of the future trajectory (M)

$$\Delta t_{\text{usage}} \approx 0.0093 * M + 0.1862 \text{ (ms)} \quad (17)$$

6 Conclusions and Future Work

In conclusion, this article proposes a pose control algorithm for omnidirectional robots in extension of our previous work [19]. The main goal of the control algorithm was being able to control the robot using only PD and FF controllers limiting the required computation resources (relative to other more complex pose control methods such as fuzzy control or MPC algorithms) while having predictive characteristics due to the derivatives required for the FF controllers.

The type of trajectory for the control algorithm was defined as a set of points described in space and time. This definition was not considered to be a limiting factor due to trajectories such as parametric or arcs being always possible to be defined as a set of points. The algorithm does not focus on generating the trajectory, although it was define as desirable at least continuity on the linear velocity's module. However, continuity on the first and second derivatives of the robot's pose are desirable to avoid step changes in the references of velocity and acceleration, respectively. Conventional PI controllers with dead zone reduction (using nonlinear Hammerstein blocks) were implemented for the angular speed control of the wheels' motors. Then, PD and FF controllers are used for pose control considering the trajectory on the local coordinate frame of the robot. The derivatives of the FF controllers were obtained by approximating a subset of points of the trajectory giving predictive characteristics to the proposed control scheme. In order to reduce the computation resources, it was defined that the set of points of the trajectory should be equally distant in time for allowing the offline computation of the linear least-squares solutions for a certain size of the future trajectory subset.

Experiments with a simulated robot and the three-wheeled omnidirectional robot of the 5DPO [14] MSL robot soccer team were performed on 3 different trajectories: a squared trajectory and an 8-shaped one with a 0º constant orientation, and a squared trajectory with variable orientation along it. The simulation experiments focused on studying the impact of the size of the future trajectory subset and a comparison with PD-only pose control scheme. It was demonstrated that a PD-only for pose control leads to the existence of a delay between the reference and the actual pose of the robot (delays of 0.20s for position and 0.24s for orientation control variables). The real experiments validated the study of the future trajectory's size and evaluated the processing time usage of the proposed controller. The conclusion was that the optimal value for the future's size should consider a time window (defines which points of the trajectory belong to the future's subset at each time instant) of 0.64s. As for the time usage of the controller, the maximum time obtained was 0.562ms representing a 1.41% percentage of the pose control period (40ms) demonstrating that the proposed controller could be used for low-cost applications (in terms of computation resources) and systems with fast dynamics.

Therefore, we consider that the proposed control algorithm for omnidirectional robots accomplished its main goals. The next developments will be studying an adaptation of the proposed control scheme for other steering geometries such as differential or tricycle robots that have nonholonomic kinematic constraints.

Declarations

Author Contributions Ricardo B. Sousa: conceptualization, methodology, formal analysis and investigation, data acquisition, software, writing – original draft preparation. Paulo G. Costa: methodology, data acquisition, software, writing – review and editing, resources. Héber Miguel Sobreira: data acquisition, software, writing – review and editing, supervision, resources. António Paulo Moreira: conceptualization, methodology, funding, software, writing – review and editing, supervision, resources.

Funding This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020. This work is also financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within scholarship 2021.04591.BD.

Availability of data and material Not applicable (this article does not have any data publicly available).

Code availability Not applicable (this article does not have any code publicly available).

Ethics approval Not applicable (this article does not contain any studies with human participants or animals performed by any of the authors).

Consent for publication All authors have approved the manuscript and agreed with its publication on the Journal of Intelligent & Robotic Systems.

Consent to participate Not applicable (this article does not contain any studies with human participants or animals performed by any of the authors).

Conflict of interest/Competing interests Not applicable (the authors have no financial or proprietary interests in any material discussed in this article).

References

1. Abiyev, R.H., Akkaya, N., Günsel, I.S.: Control of omnidirectional robot using Z-number-based fuzzy system. IEEE Transactions on Systems, Man, and Cybernetics: Systems **49**(1), 238–252 (2019). https://doi.org/10.1109/TSMC.2018.2834728
2. Abiyev, R.H., Günsel, I.S., Akkaya, N., Aytac, E., Çağman, A., Abizada, S.: Fuzzy control of omnidirectional robot. Procedia Computer Science **120**, 608–616 (2017). https://doi.org/10.1016/j.procs.2017.11.286
3. Bugeja, M.K., Fabri, S.G., Camilleri, L.: Dual adaptive dynamic control of mobile robots using neural networks. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **39**(1), 129–141 (2009). https://doi.org/10.1109/TSMCB.2008.2002851
4. Chien, I.L.: IMC-PID controller design - An extension. IFAC Proceedings Volumes **21**(7), 147–152 (1988). https://doi.org/10.1016/S1474-6670(17)53816-1
5. Costa, P., Gonçalves, J., Lima, J., Malheiros, P.: SimTwo realistic simulator: a tool for the development and validation of robot software. Theory and Applications of Mathematics & Computer Science **1**(1), 17 (2011). http://hdl.handle.net/10198/4117
6. Huang, H.C., Wu, T.F., Yu, C.H., Hsu, H.S.: Intelligent fuzzy motion control of three-wheeled omnidirectional mobile robots for trajectory tracking and stabilization. In: 2012 International conference on Fuzzy Theory and Its Applications (iFUZZY2012), pp. 107–112 (2012). https://doi.org/10.1109/iFUZZY.2012.6409684
7. Jianbin, W., Jianping, C.: An adaptive sliding mode controller for four-wheeled omnidirectional mobile robot with input constraints. In: 2019 Chinese Control And Decision Conference (CCDC), pp. 5591–5596 (2019). https://doi.org/10.1109/CCDC.2019.8832894
8. Klančar, G., Blažič, S., Zdešar, A.: C2-continuous path planning by combining Bernstein-Bézier curves. In: Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics - Volume 2: ICINCO, pp. 254–261 (2017). https://doi.org/10.5220/0006406602540261
9. Li, Z., Deng, J., Lu, R., Xu, Y., Bai, J., Su, C.Y.: Trajectory-tracking control of mobile robot systems incorporating neural-dynamic optimized model predictive approach. IEEE Transactions on Systems, Man, and Cybernetics: Systems **46**(6), 740–749 (2016). https://doi.org/10.1109/TSMC.2015.2465352
10. Li, Z., Wang, Y., Song, X., Liu, Z.: Neural adaptive tracking control for wheeled mobile robots. In: 2015 International Conference on Fluid Power and Mechatronics (FPM), pp. 610–617 (2015). https://doi.org/10.1109/FPM.2015.7337188
11. Liu, Y., Zhu, J.J., Williams, R.L., Wu, J.: Omni-directional mobile robot controller based on trajectory linearization. Robotics and Autonomous Systems **56**(5), 461–479 (2008). https://doi.org/10.1016/j.robot.2007.08.007
12. Masmoudi, M.S., Krichen, N., Masmoudi, M., Derbel, N.: Fuzzy logic controllers design for omnidirectional mobile robot navigation. Applied Soft Computing **49**, 901–919 (2016). https://doi.org/10.1016/j.asoc.2016.08.057
13. Mu, J., Yan, X.G., Jiang, B., Spurgeon, S.K., Mao, Z.: Sliding mode control for a class of nonlinear systems with application to a wheeled mobile robot. In: 2015 54th IEEE Conference on Decision and Control (CDC), pp. 4746–4751 (2015). https://doi.org/10.1109/CDC.2015.7402959
14. Nascimento, T.P., Pinto, M.A., Sobreira, H.M., Guedes, F., Castro, A., Malheiros, P., Pinto, A., Alves, H.P., Ferreira, M., Costa, P., Costa, P.G., Souza, A., Almeida, L., Reis, L.P., Moreira, A.P.: 5DPO'2011: Team description paper. In: no. Robocup (2011)
15. Petrinec, K., Kovacic, Z.: The application of spline functions and Bezier curves to AGV path planning. In: Proceedings of the IEEE International Symposium on Industrial Electronics, 2005. ISIE 2005., vol. 4, pp. 1453–1458 (2005). https://doi.org/10.1109/ISIE.2005.1529146
16. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y., et al.: Ros: an open-source robot operating system. In: ICRA workshop on open source software, vol. 3, p. 5 (2009)
17. Rossomando, F.G., Soria, C.M.: Identification and control of nonlinear dynamics of a mobile robot in discrete time using an adaptive technique based on neural PID. Neural Computing and Applications **26**, 1179–1191 (2015). https://doi.org/10.1007/s00521-014-1805-8
18. Siegwart, R., Nourbakhsh, I.R., Scaramuzza, D.: Introduction to autonomous mobile robots", 2 edn. The MIT Press, Cambridge, Massachusetts (2011)
19. Sousa, R.B., Costa, P.G., Moreira, A.P.: A pose control algorithm for omnidirectional robots. In: 2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 91–96 (2021). https://doi.org/10.1109/ICARSC52212.2021.9429803
20. Wang, C., Liu, X., Yang, X., Hu, F., Jiang, A., Yang, C.: Trajectory tracking of an omni-directional wheeled mobile robot using a model predictive control strategy. Applied Sciences **8**(2), 231 (2018). https://doi.org/10.3390/app8020231
21. Watanabe, K., Shiraishi, Y., Tzafestas, S.G., Tang, J., Fukuda, T.: Feedback control of an omnidirectional autonomous platform for mobile service robots. Journal of Intelligent and Robotic Systems **22**, 315–330 (1998). https://doi.org/10.1023/A:1008048307352

Ricardo B. Sousa obtained a M.Sc. degree in electric and computers engineering at Faculty of Engineering of the University of Porto (FEUP), in 2020. He is currently working towards the Ph.D. degree in electrical and computer engineering with FEUP, and he has been developing his reseach within the Centre for Robotics in Industry and Intelligent Systems at INESC TEC. His research interests include robotics, sensor fusion, and localisation and mapping for autonomous robots.

Paulo G. Costa received the M.Sc. and Ph.D. in Electrical and Computer Engineering on Faculty of Engineering of the University of Porto (FEUP), Portugal in 95 and 2000. He joined FEUP in 1992, and currently he is a Professor in the Electrical Engineering Department. He is also a senior researcher in Centre for Robotics in Industry and Intelligent Systems at INESC TEC. He has published more than a hundred papers in international scientific journals and conference proceedings. In addition, he participated and organized many autonomous mobile robotics competitions. Moreover, his research interests are in the field of robotics and automation: simulation, path planning, artificial vision, mobile robot localization and navigation, obstacle avoidance and perception.

Héber Miguel Sobreira was born in Leiria, Portugal, in July 1985. He graduated with an M.Sc. degree (2009) and a Ph.D. degree (2017) in Electrical Engineering from the University of Porto. Since 2009, he has been developing his research within the Centre for Robotics in Industry and Intelligent Systems at INESC TEC. His main research areas are navigation and control of indoor autonomous vehicles.

António Paulo Moreira graduated with a degree in electrical engineering at the University of Oporto, in 1986. Then, he pursued graduate studies at University of Porto, obtaining a M.Sc. degree in electrical engineering – systems in 1991 and a Ph.D. degree in electrical engineering in 1998. Presently, he is Associate Professor with tenure at the Faculty of Engineering of the University of Porto and researcher and head of the Centre for Robotics in Industry and Intelligent Systems at INESC TEC. His main research interests are process control and robotics.