A Pose Control Algorithm for Omnidirectional Robots

Removed for blind revision

***Abstract*—The pose control (position and orientation) of a robot is important to control how and when the robot gets to the desired pose at the desired time in order to perform some task. Controlling omnidirectional robots is of great interest due to their complete maneuverability. So, we use Proportional-Integrative (PI), Proportional-Derivative (PD), and Feed-Forward (FF) controllers to control the pose of an omnidirectional robot in space and in time. The proposed controller approximates the future trajectory (a subset of points) on parametric polynomials for computing the derivatives needed in the FF. In the simulations performed, it was analyzed the size of the future trajectory horizon for the controller depending on the robot's velocity, and the proposed controller was compared to PD-only and a generic GoToXY controller. The results demonstrated that the proposed controller achieves better results than the other two both in space and in time.**

***Index Terms*—control, omnidirectional robots, pose control, motion control**

I. Introduction

In mobile robotics, it is not only important for the robot to reach the desired pose (position and orientation) but also how and when the robot gets there. The best case is when the robot can follow any trajectory through its workspace of poses. Omnidirectional robots are of great interest for complete maneuverability. These robots are capable of moving in all directions at any time being suitable for dynamic environments. The translation and rotation components of this steering geometry are independent of each other. One popular use of omnidirectional robots is in robot soccer games of the RoboCup competition and, nowadays, these robots are used more and more in industrial applications [1], [2].

The control of omnidirectional robots is very important to control the pose of the robot overtime. Several works were proposed characterizing the nonlinearity and uncertain factors of these robots. These approaches are related to adaptive control [3], fuzzy control [2], [4], [5], sliding mode control [3], [6], and neural networks [7]. However, the consideration of nonlinearities increases the complexity of implementing these works. Other more classical approaches are based on the use of PI, PD, and/or PID controllers [8], [9]. A disadvantage of only using these controllers is the overshoot depending on the robot's dynamics when sudden changes happen in the position reference. Another disadvantage is the existence of a possible delay between the reference and actual pose of the robot.

This work is intended for controlling the pose of omnidirectional robots given a set of points in space with timestamps (position, orientation, and time) associated with each one. The pose control uses PI controllers for the angular speed of the wheels, PD for positioning control, and Feed-Forward (FF) controllers to mitigate the delay between the robot's reference and actual pose when using only PD controllers. Also, the pose control uses a subset of points from the desired trajectory as future poses of the robot to compute the derivatives for the FF controllers. The simulation results demonstrate that the proposed controller leads to not only the robot following the trajectory on the desired time instants while predicting sudden changes in the pose's reference of the robot.

The paper is organized as follows. Section II defines the considerations assumed in this work when defining a trajectory for the robot. Section III formulates a classical approach to pose control using only PI and PD controllers. Section IV proposes the use of FF controllers and defines the formulation of the controllers' derivatives. Section V analyses the simulation results obtained from the experiments made. Finally, Section VI presents the conclusions and future work.

II. Trajectory Definition

A. Type of trajectory

Normally, a planner of trajectories defines these as a set of points, parametric splines, or polynomials. In this paper, we considered a trajectory T as a set of points P_k defined both in space (position and orientation of the robot) and in time: $T = \{P_0(t_0), P_1(t_1), \ldots, P_{N-1}(t_{N-1})\}$, where $P_k(t_k) = \{P_{X,k}, P_{Y,k}, P_{\theta,k}\}(t_k)$. Note that even if it is used, e.g., parametric splines, these curves can be discretized to obtain a set of points with timestamps defined for each point.

The set of points in time can be characterized in three different ways. The most general alternative is defining the points without any restrictions in space or in time. Another approach is maintaining a certain distance between consecutive points. The one used in this paper is restricting the time interval as a constant value between consecutive points. However, the proposed controller works also with the first two approaches.

B. Coordinate frame of the trajectory

A trajectory could be relative to the world's coordinate frame ($\{X^W, Y^W\}$) or to the robot's local frame ($\{X^R, Y^R\}$). These two different alternatives are illustrated in figure 1 for the point P_k, where P_k^W represents the point's desired pose relative to the world and P_k^R to the robot. The orientation matrix $R^{-1}(\theta_{rob}^W)$ defined in equation 1 ($\cos(\theta_{rob}^W)$ and $\sin(\theta_{rob}^W)$ are represented by $c_{\theta_{rob}^W}$ and $s_{\theta_{rob}^W}$, respectively) is the orientation of the world relative to the robot's frame depending on the robot's current orientation (θ_{rob}^W). Then, the

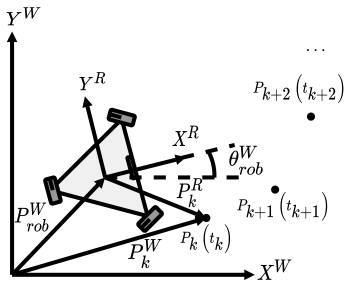


Fig. 1. Trajectory relative to the world and robot's coordinate frames

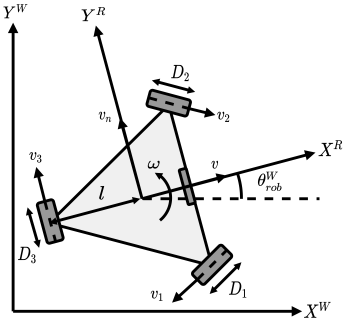


Fig. 2. Three-wheeled omnidirectional robot

transformation of the trajectory from the world to the robot's local frame can be defined as in equation 2.

$$R^{-1}(\theta^W_{rob}) = \begin{bmatrix} \mathrm{c}_{\theta^W_{rob}} & \mathrm{s}_{\theta^W_{rob}} & 0 \\ -\mathrm{s}_{\theta^W_{rob}} & \mathrm{c}_{\theta^W_{rob}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$\begin{aligned} \begin{bmatrix} P^R_{X,k} \\ P^R_{Y,k} \\ P^R_{\theta,k} \end{bmatrix} &= R^{-1}(\theta^W_{rob}) \cdot \left[\begin{bmatrix} P^W_{X,k} \\ P^W_{Y,k} \\ P^W_{\theta,k} \end{bmatrix} - \begin{bmatrix} X^W_{rob} \\ Y^W_{rob} \\ \theta^W_{rob} \end{bmatrix} \right] \\ &= \begin{bmatrix} (P^W_{X,k} - X^W_{rob}) \cdot \mathrm{c}_{\theta^W_{rob}} + (P^W_{Y,k} - Y^W_{rob}) \cdot \mathrm{s}_{\theta^W_{rob}} \\ (-P^W_{X,k} + X^W_{rob}) \cdot \mathrm{s}_{\theta^W_{rob}} + (P^W_{Y,k} - Y^W_{rob}) \cdot \mathrm{c}_{\theta^W_{rob}} \\ P^W_{\theta,k} - \theta^W_{rob} \end{bmatrix} \end{aligned} \tag{2}$$

One advantage of the trajectory being relative to the robot's local frame is that ${P_k}^R$ is the pose error relative to the current robot's pose. Moreover, the motion of omnidirectional robots can be controlled independently in terms of translation and rotation [1], [2]. The translation part is possible to decouple in the directions of X^R and Y^R. Thus, $P^R_{X,k}$, $P^R_{Y,k}$, and $P^R_{\theta,k}$ represent the error components in the direction of X^R, Y^R, and on the orientation of the robot.

III. A Classic Approach for Pose Control

In this section, it is proposed a pose control system for omnidirectional robots (the three-wheeled robot used in the simulation is illustrated in figure 2) using PI controllers for the angular speed of the motors and PD controllers for the robot's pose relative to its own frame.

A. Inverse kinematics

First, it is necessary to characterize the inverse 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) velocity desired for the robot, equation 3 computes the linear velocity (v_i) of each wheel i (considering also the distance l between the wheels and the robot's geometric center) [1]. Then, the angular speed of each motor ($\dot{\varphi}_i$) is computed depending on the gear's reduction ratio (n) and on the diameter of the wheels (D_i), as illustrated in equation 4.

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

Algorithm 1: Hammerstein Nonlinear Block

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

```
if V_in,i > V_d,i then
    V_mot,i = (V_in,i − V_d,i) + V_0,i
else if V_in,i > −V_d,i then
    if V_d,i ≠ 0 then V_mot,i = V_in,i · V_0,i / V_d,i
    else V_mot,i = 0
else
    V_mot,i = (V_in,i + V_d,i) − V_0,i
```

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

B. Angular speed control of the wheels

Next, the angular speed of the motors ($\dot{\varphi}_i$) can be controlled using a PI controller to set the motors' input voltage. This controller implemented in the simulation was tuned using the Internal Model Control (IMC) method [10]. Given experiments performed with a real robot and in simulation, the system can be considered as a first-order system. So, the IMC method [10] requires that it is estimated the output gain ($K_{p,\dot{\varphi}}$), the time constant ($\tau_{\dot{\varphi}}$), and the lag ($L_{\dot{\varphi}}$). The PI parameters (the proportional gain $K_{c,\dot{\varphi}}$ and the integration time $T_{I,\dot{\varphi}}$) are computed using equation 5, given a desired time constant for the closed-loop ($\tau_{cl,\dot{\varphi}}$).

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

Considering that the motors have a dead zone, a nonlinear block from an Hammerstein model was used to compensate for the existence of this zone. If the wheel/motor (with the robot on the ground) starts to rotate at a certain voltage V_0, V_d would be the new dead zone. The nonlinear block implemented in the controller for all the motors is described by the algorithm 1.

Finally, the windup effect is compensated by limiting the voltage computed from the PI controllers and the Hammerstein block to the maximum value supported by the motors. When the voltage exceeds these limits, the integration part of the PI controller remains unchanged.

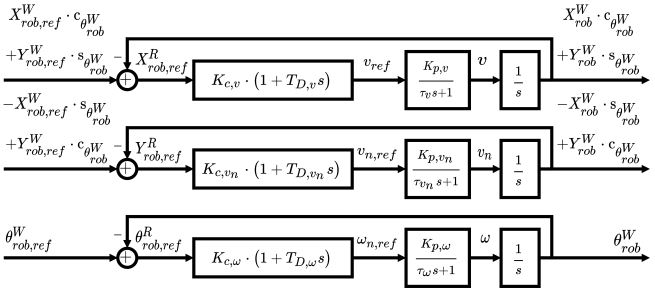


Fig. 3. PD controllers for the robot's pose

C. Position control

Similarly to the angular speed control, the evolution of v/v_n/ω relative to v_{ref}/$v_{n,ref}$/ω_{ref} resembles a first-order system (illustrated in figure 3). The integration of v, v_n, and ω estimates the robot's pose on a coordinate frame aligned with the local frame but with the same origin as the world frame. The error of the reference relative to the actual value of each component illustrated in figure 3 is the reference for the robot's pose on its local frame. So, the control of v_{ref}, $v_{n,ref}$, and ω_{ref} with a PD controller is independent from each other.

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

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

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

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

However, as it is analyzed in section V, the use of only PD controllers for pose control lead to a delay between the reference and the actual value of the controlled variables.

IV. Pose Control with Feed-Forward

Therefore, we propose the use of Feed-Forward (FF) controllers to eliminate the delay between the reference and the actual value of the control variables. This controller does not affect the stability of the system. Also, it has predictive characteristics due to the requirement of the references' first and second derivatives. The proposed control system for the omnidirectional robot is illustrated in figure 4.

A. Feed-forward definition

Equation 9 defines the transfer function of a FF controller for the systems shown in figure 3. Note that the output of each FF controllers adds to the outputs of the PD controllers to compute the references for the linear (v_{ref} and $v_{n,ref}$) and angular (ω_{ref}) velocities of the robot.

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

B. Computation of the derivatives

Given that our trajectory T is a set of points ($T = \{P_0(t_0), P_1(t_1), \ldots, P_{N-1}(t_{N-1})\}$), we already know the future poses desired for the robot. Consequently, the derivatives could be estimated considering these points in space and in time by approximating the set of points into parametric polynomials. Also, as already mentioned in section II, the time interval between consecutive points is considered to be constant. In order to normalize this time interval, we assume that it is 1 on a time domain u. The transformation $t \rightarrow u$ is characterized by equation 10.

$$u = K_T \cdot t \tag{10}$$

So, first, we define the future trajectory as a subset of points F (where $F = \{P_{l+0}(u_0), \ldots, P_{l+M-1}(u_{M-1})\}$ and $F \subset T$) with M elements on the time domain u. These elements represent the future poses of the robot on the world frame relative to the current one. Next, the second derivative ($\ddot{f}_h$ where $h = X^W_{rob,ref}, Y^W_{rob,ref}, \theta^W_{rob,ref}$), i.e., the 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 and velocity 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 $a_{h,2}$ (equivalent to the robot's acceleration).

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

$$\text{if } u = 0, 1, ..., M-1 \text{ then } \begin{cases} f_h(u_0) = a_{h,0} \\ \dot{f}_h(u_0) = a_{h,1} \end{cases} \begin{cases} a_{h,0} = P_{h,l+0} \\ a_{h,1} \approx P_{h,l+1} - P_{h,l+0} \end{cases} \tag{12}$$

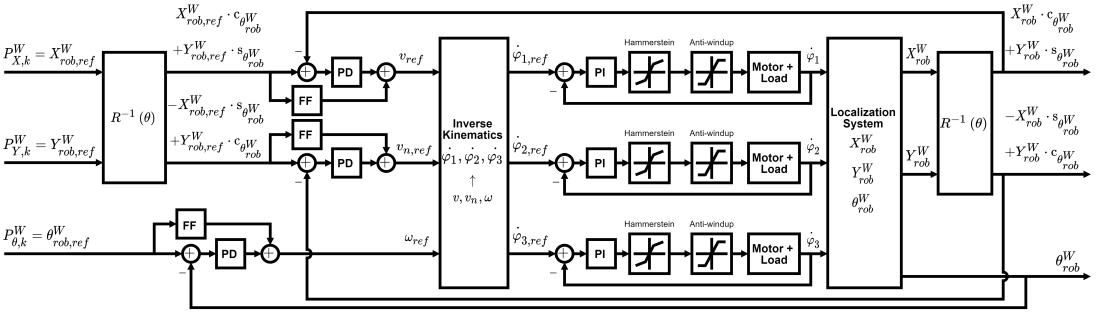


Fig. 4. Proposed trajectory controller for omnidirectional robots

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

Next, we estimate the first derivative ($\dot{g}_h$) with a first-order approximation polynomial illustrated in equation 14. The coefficients of the first-order are estimated using the least-squares algorithm, as illustrated in equation 15. An important note for both first and second-order approximations is that the orientation of F must be unwrapped to compute the linear approximations (i.e., without discontinuities).

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

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

In figure 5, it is possible to visualize the difference between approximating a future trajectory ($M = 7$), a square corner, by a second-order polynomial with and without the restrictions of initial position and velocity. First, analyzing the approximations for the first point, it is clear 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 velocity and acceleration vectors estimated with the proposed approach predicts correctly the behavior of the reference. On the robot's arrival 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

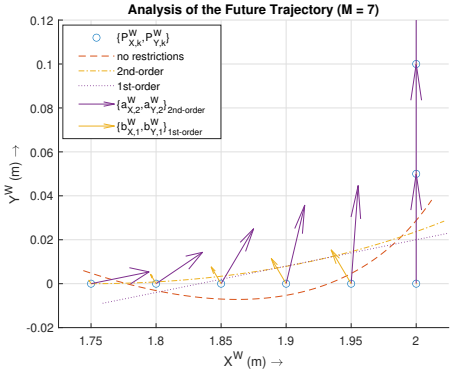


Fig. 5. Analysis of the future trajectory in XY on a square corner

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 figure 5 is similar for the robot's orientation, but only on one dimension.

Lastly, if K_T is different from 1, the derivatives should be multiplied by this scalar. Note that if we have points with a distance and a time interval between them of 0.05m and 1s, respectively, the velocity required from the robot is in average 0.05m/s with $K_T = 1$. For example, with $K_T = 2$, the velocity is increased by the same factor as K_T, i.e., the average will be 0.10m/s. Also, the derivatives $\ddot{f}_h$ and $\dot{g}_h$ are relative to the world frame. So, these derivatives must be multiplied by the matrix $R^{-1}(\theta^W_{rob})$ (equation 1) for the FF controllers formulated in this section correspond to the ones in figure 4.

V. Simulation Results

The experiments were performed in the simulator SimTwo [11] that implements the ODE physics engine, and it has available the simulation of omnidirectional robots. The robot's model used in this paper was based on the three-wheeled omnidirectional robot of a Middle Size League robot soccer team. The parameters of the motors and the robot retrieved from experiments with the real robot, and the ones defined for the PI and PD controllers are the following ones:

- characteristics of the robot:
  - $n = 12$; $C_e = 1024$ ppr

  - $l = 0.195$m; $D = 0.102$ m
- motors:
  - $K_{p,\dot{\varphi}} = 2.6181$ rad.s^{-1}.V^{-1} (relative to the wheel), $\tau_{\dot{\varphi}} = 0.198$ s, $L_{\dot{\varphi}} = 0$ s
  - $K_{c,\dot{\varphi}} = 0.57293$ V.s.rad^{-1} (relative to the wheel), $T_{I,\dot{\varphi}} = 0.19831$ s
- robot $[v, v_n, \omega]$:
  - $K_p = [1, 1, 1]$
  - $\tau = [0.129, 0.128, 0.099]$ s
  - $T_{sett.} = [0.8, 0.8, 0.8]$ s
  - $K_c = [4.41721, 4.38288, 3.40473]$ s^{-1}
  - $T_D = [0.06969, 0.06792, 0.00237]$ s

In terms of analyzing the results from the simulations performed, the maximum (max) and the average (avg) of the following three quality measures are considered:

- ε_d: distance error over time;
- ε_θ: absolute orientation error over time;
- ε_t: trajectory error (distance of the robot's position to the closest point of the desired trajectory).

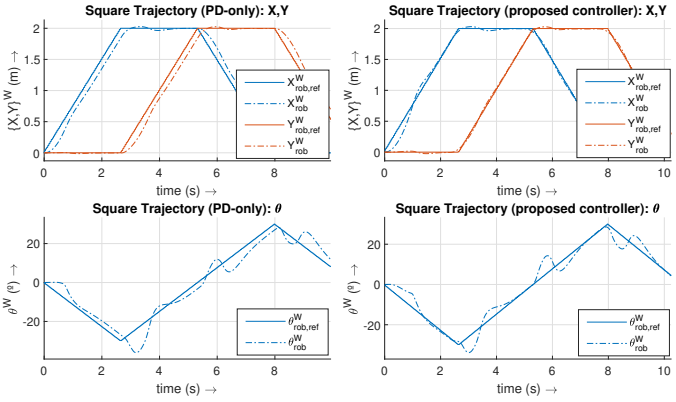
Next, it is presented three different analyses: size's definition of the future trajectory's size (M), comparison of only using PD controllers to the proposed controller in this paper, and a comparison between the proposed controller and a generic velocity controller GoToXY (follows the trajectory with a nominal velocity).

A. Size of the future trajectory (M)

The size M was studied putting the robot through a 2mx2m square with $\theta^W_{rob} = 0$º. The analysis focused on evaluating the maximum and average of the quality measures on the first corner (the interval between 0.5m before and after the corner). Another quality measure defined for this specific path is the overshoot ($o.s.$) for the trajectory outwards. As for deciding the best value for M, it is evaluated the sum of all quality measures ($\sum_{\text{all}}^{\text{m,rad}} \varepsilon_e$, and the ones related to the orientation are converted to radians, due to order of magnitude purposes) and the sum of the maximum of these quality measures ($\sum_{\text{max}}^{\text{m,rad}} \varepsilon_e$). Table I presents the results of the analysis.

Analyzing table I, the first observation is that higher M leads to a lower $o.s.$ because the robot can predict the corner earlier. With an average of 1 m/s, $\varepsilon_{max,t}$ increases with M because the prediction of the corner causes a cut inside. In the cases of 0.5 and 0.75 m/s with $M = 3$ and 4 respectively, it noted a shift in the previous claim because $o.s.$ becomes the maximum trajectory error with smaller values of M.

As for choosing the appropriate M depending on the robot's velocity, we can analyze $\sum_{\text{all}}^{\text{m,rad}} \varepsilon_e$ and $\sum_{\text{max}}^{\text{m,rad}} \varepsilon_e$. The analysis of these two quality measures (selecting the value of M that leads to the minimum value of these two measures) lead to a middle ground between predicting the corner and ε_d and ε_t. For 0.5 m/s, $M = 5$ results in the minimum value of ε_d (0.0339 m) and ε_t (0.0239 m) while decreasing 64.1% $o.s.$ relative to $M = 3$. For 0.75 m/s, $M = 7$ leads to the minimum value of ε_d (0.0552 m) and decreases $o.s.$ by 78% relative to $M = 4$. Finally, even though for 1 m/s $M = 10$ does not lead to the lowest values of ε_d and ε_t, the difference is less than 0.01 m to the minimum value (0.01m in a 2m x 2m square is 0.05%) while decreasing $o.s.$ in 38% relative to $M = 8$. So, we propose setting M as 5, 7, and 10 for 0.5, 0.75, and 1 m/s, respectively, for the simulated robot based on the results presented in table I.



(a) (b)

Fig. 6. Comparison of the proposed controller with a PD-only control

B. Comparison with PD-only position control

Figure 6 illustrated the comparison of the proposed controller to only using PD controllers for the robot's pose on a square trajectory with a changing orientation over time for the robot. The main disadvantage of using only PD controllers is observable in 6a as the actual pose of the robot is delayed relative to the desired one. Indeed, the delay of approximately 0.23 s for $\{X, Y\}^W$ and 0.3 s for θ^W leads to a $\varepsilon_{max,d}$ and a $\varepsilon_{max,\theta}$ of 0.2247 m and 12.677º, respectively (compared to 0.0583 m and 8.497º for the proposed controller). However, $\varepsilon_{max,t}$ is still similar to the proposed controller: 0.0360 m versus 0.0460 m, respectively.

C. Comparison with GoToXY

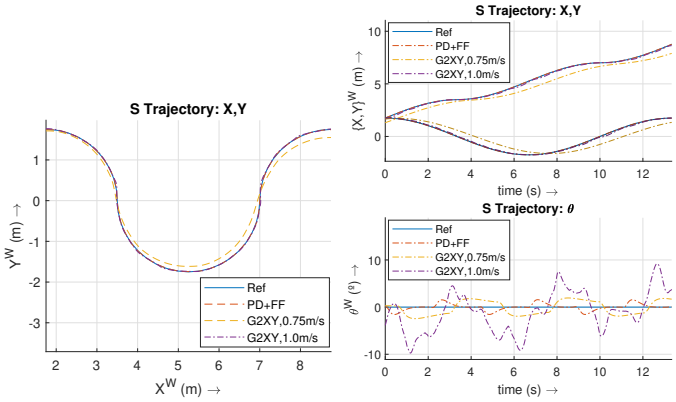
Lastly, a comparison of the proposed controller with a generic GoToXY is illustrated in figure 7 on a S-type trajectory with an average velocity set by the reference of 0.75 m/s. With the same velocity set for the GoToXY, the controller cannot "catch" the reference. The main reason is that GoToXY does not compensate for the initial delay created by the robot starting from a standstill resulting in a 0.9697 m, 2.444º, and 0.1584 m for $\varepsilon_{max,d}$, $\varepsilon_{max,\theta}$, and $\varepsilon_{max,t}$, respectively. With the nominal velocity at 1.0 m/s, GoToXY achieves a $\varepsilon_{max,d}$, $\varepsilon_{max,\theta}$, and $\varepsilon_{max,t}$ of 0.2028 m, 9.739º, and 0.036 m. In contrast, the proposed controller have a $\varepsilon_{max,d}$, $\varepsilon_{max,\theta}$, and $\varepsilon_{max,t}$ of 0.0359 m, 1.574º, and 0.0233 m, respectively. While the proposed controller led to the robot's pose be similar to the desired one, GoToXY introduces a spacial and time delay with the same velocity.

VI. Conclusions and Future Work

In conclusion, the proposed controller achieved its main purpose of following a trajectory composed of a set of points

TABLE I
EXPERIMENTAL RESULTS OF DIFFERENT SIZES FOR THE FUTURE TRAJECTORY ON A SQUARE CORNER

K_T	v_{avg} (m/s)	M	$\varepsilon_{avg,d}$ (m)	$\varepsilon_{avg,\theta}$ (º)	$\varepsilon_{avg,t}$ (m)	$\varepsilon_{max,d}$ (m)	$\varepsilon_{max,\theta}$ (º)	$\varepsilon_{max,t}$ (m)	$o.s.$ (m)	$\sum_{\text{all}}^{\text{m,rad}} \varepsilon_e$	$\sum_{\text{max}}^{\text{m,rad}} \varepsilon_e$
10	0.50	3	0.0224	0.822	0.0126	0.0776	2.548	0.0418	0.0418	0.2550	0.2057
		4	0.0130	0.503	0.0093	0.0405	1.655	0.0271	0.0271	0.1547	0.1236
		5	0.0108	0.532	0.0084	0.0339	1.316	0.0239	0.0150	0.1243	0.0958
		6	0.0112	0.627	0.0088	0.0485	1.628	0.0329	0.0111	0.1519	0.1209
		8	0.0173	0.628	0.0128	0.0812	1.472	0.0561	0.0092	0.2133	0.1722
15	0.75	4	0.0556	2.516	0.0292	0.1467	10.281	0.0682	0.0682	0.5913	0.4625
		6	0.0281	3.062	0.0140	0.0749	7.865	0.0314	0.0250	0.3641	0.2686
		7	0.0188	0.914	0.0138	0.0552	2.257	0.0444	0.0150	0.2025	0.1540
		8	0.0206	0.725	0.0156	0.0730	1.439	0.0567	0.0108	0.2145	0.1656
		9	0.0257	0.573	0.0188	0.0914	1.477	0.0714	0.0095	0.2526	0.1981
20	1.00	8	0.0549	3.646	0.0241	0.1311	11.785	0.0543	0.0500	0.5837	0.4411
		9	0.0478	2.038	0.0235	0.0924	5.264	0.0568	0.0393	0.3872	0.2804
		10	0.0442	0.978	0.0244	0.1004	1.977	0.0693	0.0310	0.3209	0.2352
		11	0.0445	2.193	0.0254	0.1106	3.712	0.0764	0.0223	0.3823	0.2741
		12	0.0460	1.847	0.0280	0.1253	2.878	0.0878	0.0180	0.3876	0.2813



(a) (b)

Fig. 7. Comparison of the proposed controller with GoToXY

(each point sets with a desired position and orientation for the robot at a certain time instant) in both space and time. Furthermore, it has predictive characteristics that avoid overshoots with sudden changes in the robot's motion direction. As for comparisons with a PD-only controlling system and a generic GoToXY (sets a nominal velocity), the proposed controller led to improvements for following the trajectory in space and time. As future work, the proposed controller adapted to consider possible limitations in terms of maximum angular speed of the wheels, optimizing the parameterization of the PI and PD controllers, comparison in terms of performance and computational requirements to Model Predictive Controllers (MPC), and it will be performed tests with real robots.

REFERENCES

[1] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*, 2nd ed. Cambridge, Massachusetts: The MIT Press, 2011.

[2] R. H. Abiyev, I. S. Günsel, N. Akkaya, E. Aytac, A. Çağman, and S. Abizada, "Fuzzy control of omnidirectional robot," *Procedia Computer Science*, vol. 120, pp. 608–616, Dec. 2017.

[3] W. Jianbin and C. Jianping, "An adaptive sliding mode controller for four-wheeled omnidirectional mobile robot with input constraints," in *2019 Chinese Control And Decision Conference (CCDC)*, June 2019, pp. 5591–5596.

[4] M. S. Masmoudi, N. Krichen, M. Masmoudi, and N. Derbel, "Fuzzy logic controllers design for omnidirectional mobile robot navigation," *Applied Soft Computing*, vol. 49, pp. 901–919, Dec. 2016.

[5] R. H. Abiyev, N. Akkaya, and I. Gunsel, "Control of omnidirectional robot using z-number-based fuzzy system," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 1, pp. 238–252, Jan. 2019.

[6] J. Mu, X. Yan, B. Jiang, S. K. Spurgeon, and Z. Mao, "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)*, Dec. 2015, pp. 4746–4751.

[7] Z. Li, Y. Wang, X. Song, and Z. Liu, "Neural adaptive tracking control for wheeled mobile robots," in *2015 International Conference on Fluid Power and Mechatronics (FPM)*, Aug. 2015, pp. 610–617.

[8] K. Watanabe, Y. Shiraishi, S. G. Tzafestas, J. Tang, and T. Fukuda, "Feedback control of an omnidirectional autonomous platform for mobile service robots," *Journal of Intelligent and Robotic Systems*, vol. 22, no. 3, pp. 315–330, July 1998.

[9] F. G. Rossomando and C. M. Soria, "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*, vol. 26, no. 5, pp. 1179–1191, July 2015.

[10] I.-L. Chien, "IMC-PID controller design - an extension," *IFAC Proceedings Volumes*, vol. 21, no. 7, pp. 147–152, Aug. 1988.

[11] P. Costa, J. Gonçalves, J. Lima, and P. Malheiros, "Simtwo realistic simulator: A tool for the development and validation of robot software," *Theory and Applications of Mathematics & Computer Science*, vol. 1, no. 1, pp. 17–, Apr. 2011.