Robotica (2015) volume 33, pp. 2186–2203. © Cambridge University Press 2014
doi:10.1017/S0263574714001325

Trajectory-tracking controller design with constraints in the control signals: a case study in mobile robots

Mario Emanuel Serrano†*, Gustavo Juan Eduardo Scaglia†, Fernando Auat Cheein‡, Vicente Mut§ and Oscar Alberto Ortiz†

†*Instituto de Ingeniería Química, Universidad Nacional de San Juan, San Juan, Argentina*
E-mails: gscaglia@unsj.edu.ar, rortiz@unsj.edu.ar
‡*Universidad Técnica Federico Santa María, Valparaíso, Chile*
E-mail: fernando.auat@usm.cl
§*Instituto de Automática, Universidad Nacional de San Juan, San Juan, Argentina*
E-mail: vmut@unsj.edu.ar

(Accepted April 29, 2014. First published online: May 29, 2014)

SUMMARY
This paper is a continuation of a previous work of authors, Scaglia *et al.* [G. J. E. Scaglia, L. M. Quintero, V. Mut and F. Di Sciascio, "Numerical methods based controller design for mobile robots," *Robotica* **27**(2), 269–279 (2009)]. A method is presented to choose the controller parameters such that, the values of the control actions do not exceed the maximum allowable and the tracking errors tend to zero. In addition, the analysis of the controller design parameters is included. The experimental results (laboratory experiments and a real world application) demonstrate the efficiency of the controller.

KEYWORDS: Control system design; Nonlinear model; Tracking trajectory control.

1. Introduction
In recent years there has been an increasing amount of research on mobile robotics. Mobile robots are currently used in industry, for domestic needs (vacuum cleaners, lawn mowers, pets), in difficult-to-access or dangerous places/areas (space, army, nuclear-waste cleaning) and also for entertainment (competition – robot soccer). The main problems found in mobile robot control are trajectory tracking,[1,6–8] contour following,[19–21] and path tracking.[9,10,15]

The use of trajectory tracking is justified in structured workspaces as well as in partially structured workspaces, where unexpected obstacles can be found during the navigation. In the first case, the desired trajectory can be set from a global trajectory planner. In the second case, the algorithms used to avoid obstacles usually re-plan the trajectory in order to avoid a collision, generating a new reference trajectory from this point on. In general, the objective is to find the control actions that make the mobile robot reach Cartesian position (x, y) with a pre-established orientation θ in each sampling period. These combined actions result in tracking the desired trajectory of the mobile robot. Another typical problem, covered in the literature by other authors[2,3] is the trajectory tracking with constraints in the control actions. In general, in a robot mobile system, the linear and angular velocity constraints prevent from mobile robot slipping and the saturation of actuators.

Several tracking controller designs employing a simplified linear model have been reported. In ref. [4], a controller for trajectory tracking is designed using the kinematic model of the mobile robot and a transformation matrix. Such matrix is singular if the linear velocity of the mobile robot is

* Corresponding author. E-mail: eserrano@fi.unsj.edu.ar

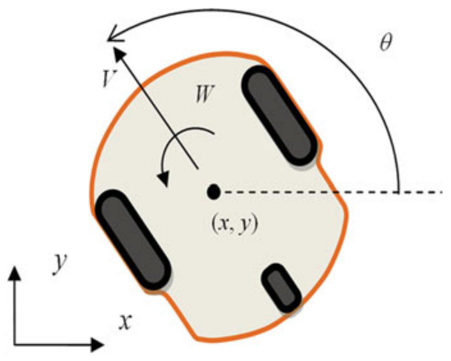

Fig. 1. Geometric description of the mobile robot.

zero; therefore, the effectiveness of this controller is only assured if the velocity is different from zero. Simulation results using linear velocity different from zero as initial condition are shown in this paper. Consequently, a more generalized approach is desirable. Nonlinear system theory has been employed to solve this problem. In ref. [2], a saturation feedback controller (where saturation constraints of the velocity inputs are incorporated into the controller design) is introduced. In ref. [5], a dynamic feedback linearization technique is used to control a mobile robot platform. Usually, the desired trajectory is obtained by using a reference virtual robot; therefore, all the kinematic constraints are implicitly considered by the desired trajectory. The control inputs are mostly obtained by a combination of feed forward inputs, calculated from desired trajectory, and feedback control law, as in ref. [5]. In ref. [3], a model-predictive trajectory-tracking control applied to a mobile robot is presented. Linearized tracking-error dynamics is used to predict future system behavior and a control law is derived from a quadratic cost function penalizing the system tracking error and the control effort.

The authors in ref. [6] present a novel linear interpolation-based methodology to design control algorithms for trajectory tracking of mobile vehicles. In such a work, it is assumed that the evolution of the system can be approximated by linear interpolation in each sampling time. A novel approach for trajectory tracking of a mobile-robot formation by using linear algebra theory and numerical methods is presented in ref. [7]. Using this strategy, in the formation of mobile robots it is possible to change its configuration (shape and size) and follow different trajectories in a precise way, minimizing the tracking and formation errors. To face up to sudden velocity changes and to improve the performance of the system, Rosales *et al.*[8] propose working with the dynamic model of the mobile robot.

In ref. [1] a novel trajectory-tracking controller is presented. The originality of this control approach is based on the application of linear algebra for trajectory tracking, where the calculations of control actions are obtained solving a system of linear equations. The methodology developed for tracking the desired trajectory (x_d and y_d) is based on determining the trajectories of the remaining state variables, thus the tracking error tends to zero. These states are determined by analyzing the conditions so that the system of linear equations has the exact solution. In order to achieve this objective, only two control variables are available: the linear velocity (V) and the angular velocity (W) of the robot (Fig. 1).

In this work, the saturation constraints in the control inputs (the linear and angular velocities) were incorporated in the controller design, to solve the problem of high values in the control actions obtained in refs. [1] and [6]. The main contribution of this paper is the application of a new methodology to find the controller parameters proposed in refs. [1] and [6]. The proposed method can be used to find values of the control actions that do not exceed the actuators saturation limits and make the tracking errors tend to zero. The methodology is based on a nonlinear programming (NLP) technique to choose the parameters of the controller in such a way that they can tackle the velocity constraints of the robot.

The paper is organized as follows: Section 2 shows the methodology for the design of the control system, using linear algebra. The analysis of the controller design parameters and the methodology for their selection are included in Section 3. Experimental results are shown in Section 4. The system performance under additive disturbances in control signals is shown in Sections 4.3 and 4.4. In addition, in Section 4.4, experimental results showing the tracking error decrease related to sampling period reduction are shown. Section 4.5 shows a real world application of control methodologies. Section 5 presents relevant discussions of the results obtained. Section 6 shows conclusions obtained in this work.

2. Methodology for Controller Design

This paper is a continuation of a previous work of the authors in refs. [1] and [9]. A nonlinear kinematic model for a mobile robot will be used as shown in Fig. 1; this model, which has been used in several recent papers of other authors such as Scaglia *et al.*,[1] Rosales *et al.*,[8] Jung *et al.*,[19] and Toiberos *et al.*,[20] is represented by Eq. (1), where V is the linear velocity of the mobile robot, W is the angular velocity of the mobile robot, (x, y) is the Cartesian position, and θ is the orientation of the mobile robot. The unicycle model is

$$\begin{cases} \dot{x} = V\cos(\theta) \\ \dot{y} = V\sin(\theta). \\ \dot{\theta} = W \end{cases} \tag{1}$$

The aim is to find the values of V and W so that the mobile robot may follow a pre-established trajectory. We assume that the mobile robot is moving on a horizontal plane without slip.

Through the Euler's approximation of the kinematic model of the mobile robot (Eq. (1)), the following set of equations is obtained:

$$\begin{cases} x_{(n+1)} = x_{(n)} + T_0 V_{(n)}\cos(\theta_{(n)}) \\ y_{(n+1)} = y_{(n)} + T_0 V_{(n)}\sin(\theta_{(n)})\ , \\ \theta_{(n+1)} = \theta_{(n)} + T_0 W_{(n)} \end{cases} \tag{2}$$

where $(x_{(n)}, y_{(n)})$ is the Cartesian position of the mobile robot at time n, $V_{(n)}$ and $W_{(n)}$ are the linear velocity and angular velocity at the instant n, respectively, and $\theta_{(n)}$ is the orientation of the mobile robot. The sample time is T_0.

This can be re-arranged as

$$\begin{bmatrix} x_{(n+1)} \\ y_{(n+1)} \\ \theta_{(n+1)} \end{bmatrix} = \begin{bmatrix} x_{(n)} \\ y_{(n)} \\ \theta_{(n)} \end{bmatrix} + T_0 \begin{bmatrix} \cos(\theta_{(n)}) & 0 \\ \sin(\theta_{(n)}) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V_{(n)} \\ W_{(n)} \end{bmatrix}. \tag{3}$$

If the desired trajectory $x_{d(n+1)}$, $y_{d(n+1)}$, and $\theta_{d(n+1)}$ is known, then $x_{(n+1)}$, $y_{(n+1)}$, and $\theta_{(n+1)}$ in Eq. (3) can be substituted by $x_{d(n+1)} - k_v(x_{d(n+1)} - x_{(n)})$, $y_{d(n+1)} - k_v(y_{d(n+1)} - y_{(n)})$, and $\theta_{ez(n+1)} - k_w(\theta_{ez(n+1)} - \theta_{(n)}$; see ref. [9]). The orientation θ_{ez} represents the necessary orientation to make the mobile robot tend to the reference trajectory; k_v and k_w are positive constants that allow us to adjust the performance of the proposed control system. Where $0 < k_v < 1$ and $0 < k_w < 1$ to the tracking, errors tend to zero, see ref. [6].

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} x_{d(n+1)} - k_v(x_{d(n)} - x_{(n)}) - x_{(n)} \\ y_{d(n+1)} - k_v(y_{d(n)} - y_{(n)}) - y_{(n)} \\ \theta_{ez(n+1)} - k_w(\theta_{ez(n)} - \theta_{(n)}) - \theta_{(n)} \end{bmatrix}, \quad B = \begin{bmatrix} \cos(\theta_{(n)}) & 0 \\ \sin(\theta_{(n)}) & 0 \\ 0 & 1 \end{bmatrix}. \tag{4}$$

Then by replacing Eq. (4) into Eq. (3), the following equation is obtained:

$$B \begin{bmatrix} V_{(n)} \\ W_{(n)} \end{bmatrix} = \frac{1}{T_0} \underbrace{\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}}_{b}. \tag{5}$$

Equation (5) is a system with three equations and two unknown variables. The optimum solution of it from mean squares is obtained from normal equation (Eq. (6); see refs. [16] and [18]).

$$B^T B \begin{bmatrix} V_{(n)} \\ W_{(n)} \end{bmatrix} = \frac{1}{T_0} B^T \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} \Rightarrow B^T B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad B^T \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} \Delta x \cos(\theta_{(n)}) + \Delta y \sin(\theta_{(n)}) \\ \Delta\theta \end{bmatrix}. \tag{6}$$

$$\begin{bmatrix} V_{(n)} \\ W_{(n)} \end{bmatrix} = \begin{bmatrix} \frac{\Delta x}{T_0} \cos(\theta_{(n)}) + \frac{\Delta y}{T_0} \sin(\theta_{(n)}) \\ \frac{\Delta\theta}{T_0} \end{bmatrix}. \tag{7}$$

In order to guarantee that the system shown in Eq. (5) has the exact solution, constants a_1 and a_2 must exist. In other words, if $b \in EC\mathbf{B}$ (column space of B), then

$$a_1 \begin{bmatrix} \cos(\theta_{(n)}) \\ \sin(\theta_{(n)}) \\ 0 \end{bmatrix} + a_2 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}; \quad a_1, a_2 \in \Re. \tag{8}$$

Thus, from Eq. (8),

$$a_1 \begin{bmatrix} \cos(\theta_{(n)}) \\ \sin(\theta_{(n)}) \\ 0 \end{bmatrix} = \begin{bmatrix} \Delta x \\ \Delta y \\ 0 \end{bmatrix} \Rightarrow \frac{\sin(\theta_{(n)})}{\cos(\theta_{(n)})} = \frac{\Delta y}{\Delta x}. \tag{9}$$

The $\theta_{(n)}$ in Eq. (9) represents the orientation that the mobile robot should have in order to ensure the exact solution of the system shown in Eq. (5). Hence, the tracking error will tend to zero and the mobile robot will be able to reach and follow the desired trajectory.[6,8,9] Following the methodology presented in ref. [9], considering Eq. (9), the desired orientation is defined by

$$\tan\theta_{ez(n)} = \frac{\sin\theta_{ez(n)}}{\cos\theta_{ez(n)}} = \frac{\Delta y}{\Delta x} = \frac{y_{d(n+1)} - k_v(y_{d(n)} - y_{(n)}) - y_{(n)}}{x_{d(n+1)} - k_v(x_{d(n)} - x_{(n)}) - x_{(n)}}, \tag{10}$$

where $\theta_{ez(n)}$ represents the necessary orientation to make the mobile robot tend to the desired trajectory. The desired orientation for the mobile robot guarantees that the nonholonomic system reaches the pre-established trajectory with a zero-error convergence.

Finally, the controller proposed for the mobile robot is given in Eq. (11):

$$\begin{bmatrix} V_{(n)} \\ W_{(n)} \end{bmatrix} = \begin{bmatrix} \frac{x_{d(n+1)} - k_v(x_{d(n)} - x_{(n)}) - x_{(n)}}{T_0} \cos(\theta_{ez(n)}) + \frac{y_{d(n+1)} - k_v(y_{d(n)} - y_{(n)}) - y_{(n)}}{T_0} \sin(\theta_{ez(n)}) \\ \frac{\theta_{ez(n+1)} - k_w(\theta_{ez(n)} - \theta_{(n)}) - \theta_{(n)}}{T_0} \end{bmatrix}. \tag{11}$$

3. Analysis of the Controller Design Parameters

In this section, the performance of the system when the controller parameters vary is analyzed. We present a methodology to find the controller parameters, so that, the control actions do not exceed the maximum allowable and the tracking errors tend to zero. These approaches solve the problem of high values in the control actions presented in refs. [1] and [6].

3.1. Analysis of parameters

Considering that the desired trajectory satisfies Eq. (2),

$$\begin{cases} x_{d(n+1)} = x_{d(n)} + T_0 V_{d(n)} \cos(\theta_{d(n)}) \\ y_{d(n+1)} = y_{d(n)} + T_0 V_{d(n)} \sin(\theta_{d(n)}) \\ \theta_{d(n+1)} = \theta_{d(n)} + T_0 W_{d(n)} \end{cases}. \tag{12}$$

Equations (13) and (14) represent the desired velocities so that the mobile robot tends to the desired trajectory at time n:

$$V_{d(n)} = \frac{x_{d(n+1)} - x_{d(n)}}{T_0}\cos(\theta_{d(n)}) + \frac{y_{d(n+1)} - y_{d(n)}}{T_0}\sin(\theta_{d(n)}), \tag{13}$$

$$W_{d(n)} = \frac{\theta_{d(n+1)} - \theta_{d(n)}}{T_0}. \tag{14}$$

It is assumed that in each sampling time is met the following: $V_{d(n)} < V_{\max}$ and $W_{d(n)} < W_{\max}$, where $V_{\max}$ and $W_{\max}$ are the maximum allowable speeds.

If $k_w \to 1^-$ and considering Eq. (10),

$$\lim_{k_w \to 1^-} \theta_{ez(n)} = a\tan\frac{y_{d(n+1)} - y_{d(n)}}{x_{d(n+1)} - x_{d(n)}} = \theta_{d(n)}. \tag{15}$$

Therefore, considering Eq. (11),

$$\lim_{\substack{k_v \to 1^- \\ k_w \to 1^-}} V_{(n)} = \frac{x_{d(n+1)} - x_{d(n)}}{T_0}\cos(\theta_{d(n)}) + \frac{y_{d(n+1)} - y_{d(n)}}{T_0}\sin(\theta_{d(n)}) = V_{d(n)}, \tag{16}$$

$$\lim_{\substack{k_v \to 1^- \\ k_w \to 1^-}} W_{(n)} = \frac{\theta_{d(n+1)} - \theta_{d(n)}}{T_0} = W_{d(n)}. \tag{17}$$

As $V_{d(n)} < V_{\max}$ and $W_{d(n)} < W_{\max}$, from Eqs. (16) and (17) it can be seen that, when $k_v \to 1^-$ and $k_w \to 1^-$, then $V_{(n)} \to V_{d(n)}$ and $W_{(n)} \to W_{d(n)}$; therefore $V_{(n)} < V_{\max}$ and $W_{(n)} < W_{\max}$. The latter indicates that a parameter value below (but close to) one, guarantees an acceptable trajectory tracking.

If $k_v = 1$ and $k_w = 1$, it verifies that

$$V_{(n)} = \frac{x_{d(n+1)} - x_{d(n)}}{T_0}\cos(\theta_{d(n)}) + \frac{y_{d(n+1)} - y_{d(n)}}{T_0}\sin(\theta_{d(n)}) = V_{d(n)}, \tag{18}$$

$$W_{(n)} = \frac{\theta_{d(n+1)} - \theta_{d(n)}}{T_0} = W_{d(n)}. \tag{19}$$

The latter demonstrates that a value below but close to one of the parameters (k_v and k_w) makes the actual velocities of the robot to be close to the one used to generate the trajectory, and the tracking error to converge to zero.

The methodology proposed to solve the problem of selecting the value of the parameters k_v and k_w is shown as follows.

3.2. *Controller parameters by nonlinear programming*

Nonlinear programming deals with the problem of optimizing an objective function in the presence of equality and inequality constraints, where some of the constraints or the objective function is nonlinear. Typically a NLP is posed as

$$\begin{array}{ll} \text{Minimize} & f(x) \\ \text{subject to} & \begin{cases} g_i(x) \le 0 & \text{for} \quad i = 1, \ldots, m \\ h_i(x) = 0 & \text{for} \quad i = 1, \ldots, p \,, \\ x \in X \end{cases} \end{array}$$

where $f, g_1, \ldots, g_m, h_1, \ldots, h_p$ are functions defined on R^n, X is a subset of R^n, and x is a vector of n components $x_1, \ldots, x_n$. The above problem must be solved for the values of the variables $x_1, \ldots,$ x_n that satisfy the restrictions and mean while minimizing the function f. The function f is usually called the objective function, or the criterion function. Each of the constraints $g_i \le 0$ for $i = 1, \ldots,$

m is called an *inequality constraint*, and each of the constraints $h_i = 0$ for $i = 1, \ldots, p$ is called an *equality constraint*.[18]

Considering the issue addressed in this paper, the conditions, for non-saturation of control actions (Eq. (11)), are given by

$$
\begin{aligned}
V_{(n)} = {} & \frac{x_{d(n+1)} - k_v(x_{d(n)} - x_{(n)}) - x_{(n)}}{T_0} \cos(\theta_{ez(n)}) \\
& + \frac{y_{d(n+1)} - k_v(y_{d(n)} - y_{(n)}) - y_{(n)}}{T_0} \sin(\theta_{ez(n)}) < V_{\max},
\end{aligned}
\tag{20}
$$

$$
W_{(n)} = \frac{\theta_{ez(n+1)} - k_w(\theta_{ez(n)} - \theta_{(n)}) - \theta_{(n)}}{T_0} < W_{\max}. \tag{21}
$$

To ensure that the robot movements tend to the desired trajectory, Eq. (10) must be satisfied.

$$
\theta_{ez(n+1)} = a \tan \frac{y_{d(n+1)} - k_v(y_{d(n)} - y_{(n)}) - y_{(n)}}{x_{d(n+1)} - k_v(x_{d(n)} - x_{(n)}) - x_{(n)}}, \tag{22}
$$

for

$$
0.92 < k_v < 1, \tag{23}
$$

$$
0.94 < k_w < 1. \tag{24}
$$

Equations (23) and (24) ensure that there are no sharp variations in the system. Such equations arise from analyzing the conditions to allow the tracking error to tend to zero, see ref. [6]. The upper limit is selected in accordance with the condition proposed by Scaglia *et al.*,[1,6] where the parameters must be lesser than one to ensure the zero errors convergence. The lower limits represent the values of k_v and k_w when the mobile robot reaches the desired trajectory. If the lower limits are close to zero the controller corrects the tracking error quickly; however, they can lead to unwanted oscillations.

Equations (16) and (17) show that when $k_v \to 1^-$ and $k_w \to 1^-$, then $V_{(n)} \to V_{d(n)}$ and $W_{(n)} \to W_{d(n)}$. The aim of this work is to reduce the k_v and k_w values without exceeding the maximum values of the control actions while guaranteeing the zero error convergence. The desired minimum values for k_v and k_w are 0.92 and 0.94 respectively. Thus, Eq. (25) defines the cost function with only one global minimum,

$$
f(k_v, k_w) = k_v^2 + k_w^2. \tag{25}
$$

With these considerations, the problem of selection of the parameters k_v and k_w is analyzed as a NLP problem. Therefore, the equations that comprise the problem of NLP that must be solved in each sampling time are

$$
\begin{aligned}
&\text{Minimize} \quad f(k_v, k_w) = k_v^2 + k_w^2 \\
&\text{subject to} \begin{cases}
\frac{x_{d(n+1)} - k_v(x_{d(n)} - x_{(n)}) - x_{(n)}}{T_0} \cos(\theta_{ez(n)}) + \frac{y_{d(n+1)} - k_v(y_{d(n)} - y_{(n)}) - y_{(n)}}{T_0} \sin(\theta_{ez(n)}) < V_{\max} \\
\frac{\theta_{ez(n+1)} - k_w(\theta_{ez(n)} - \theta_{(n)}) - \theta_{(n)}}{T_0} < W_{\max} \\
0.92 < k_v < 1 \\
0.94 < k_w < 1 \\
\theta_{ez(n)} = a \tan \frac{y_{d(n+1)} - k_v(y_{d(n)} - y_{(n)}) - y_{(n)}}{x_{d(n+1)} - k_v(x_{d(n)} - x_{(n)}) - x_{(n)}}
\end{cases}
\end{aligned}
$$

To resolve this problem of NLP in ref. [17], different algorithms based on the concept of trust-region-reflective are developed. In this paper this problem is solved by trust-region-reflective algorithm in

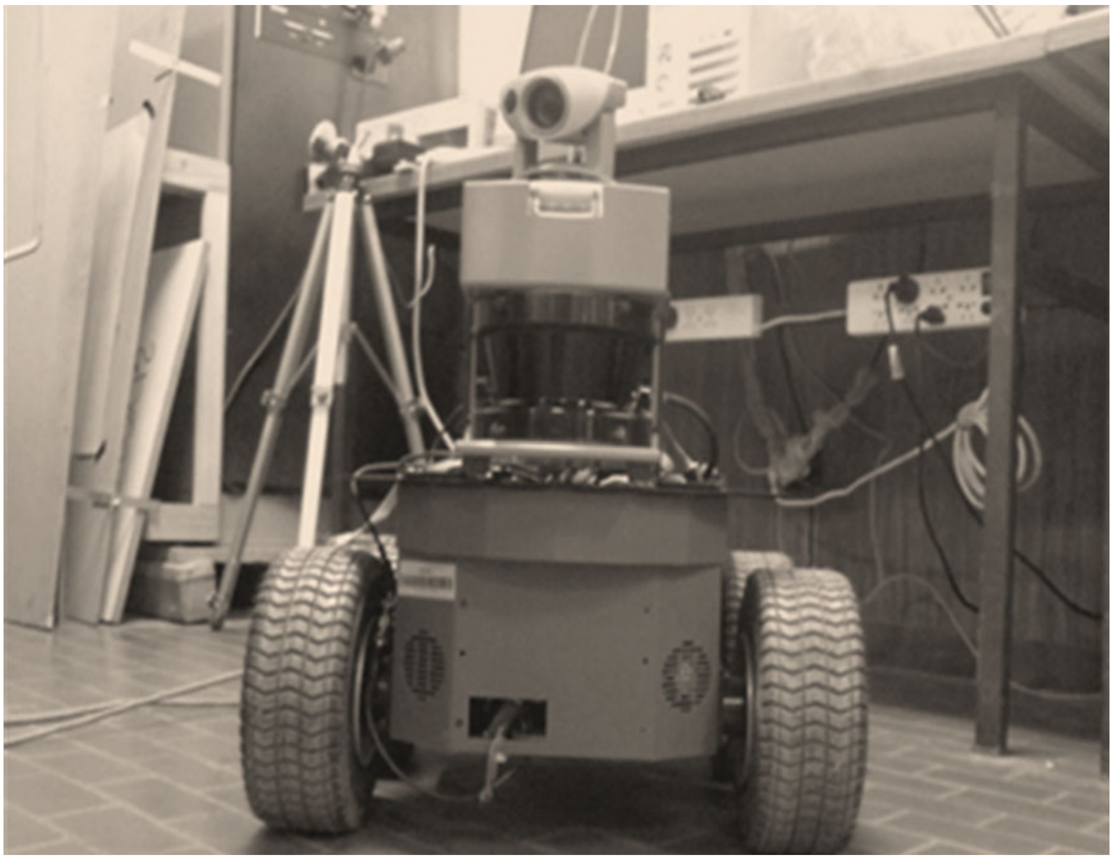

Fig. 2. PIONEER 3AT and the laboratory.

each sample time. The values k_v and k_w found are used in Eq. (11) to find the control action that will be applied in each sample time.

Remark 1: In the implementation of this work the NLP problem is solve using the "*fmincon*" of MATLAB. The function "*fmincon*" uses one of three algorithms: active-set, interior-point, or, the default option is trust-region-reflective. This function solves the NLP problem proposed in each sample time.

4. Experimental Results

To support our claims, we performed several experiments using a PIONEER 3AT mobile robot. The PIONEER 3AT mobile robot includes an estimation system based on odometric-based positioning system. Updating through external sensors is necessary. This problem is separated from the strategy of trajectory tracking and it is not considered in this paper.[10,22] The PIONEER 3AT has a proportional-integral-derivative (PID) velocity controller used to maintain the velocities of the mobile robot at the desired value. Figure 2 shows the PIONEER 3AT and the laboratory facilities where the experiments were carried out.

4.1. Experimental results: controller parameters by nonlinear programming

In order to test the proposed controller, two trajectories that satisfy Eq. (2) are generated. In the first experimentation the desired trajectory is a straight line and a circumference, the linear and angular velocities are $V_{d(n)} = 0.3$ m/s, $W_{d(n)} = 0$ rad/s, and $V_{d(n)} = 0.3$ m/s, $W_{d(n)} = 0.25$ rad/s respectively. The control actions applied at each sampling instant are given by Eq. (11), the controller parameters (k_v and k_w) are calculated online using the methodologies introduced in Section 3.

The control action constraints considered are: $V_{\max} = 0.5$ m/s and $W_{\max} = 0.45$ rad/s; the initial conditions of the desired trajectory are: $x_{d(0)} = 0.2$ m and $y_{d(0)} = 0.2$ m; and the initial conditions

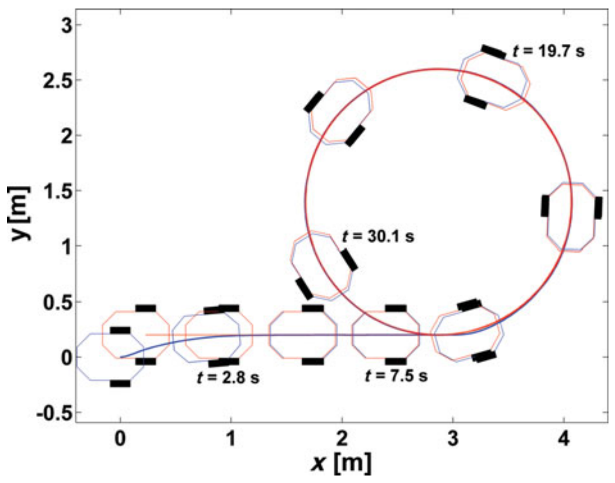

Fig. 3. Robot mobile position, experimental and desired trajectory with $T_0 = 0.1$ s. Desired trajectory: red-dot line; mobile robot position: blue line. The time is shown in some of instants together to the robots.

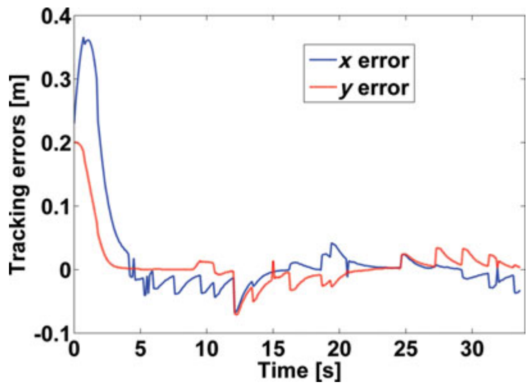

Fig. 4. Tracking errors.

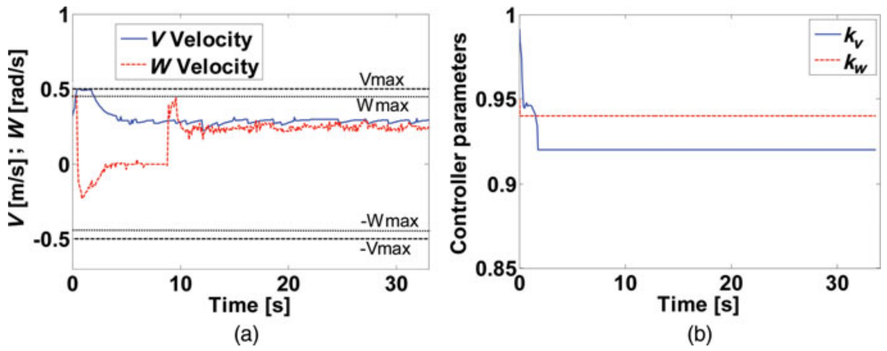

Fig. 5. (a) Control actions and the maximum allowable values. (b) Controller parameters selected using nonlinear programming techniques.

for the mobile robot are:

$$[\, x_{(0)} \;\; y_{(0)} \;\; \theta_{(0)} \,] = [\, 0 \; 0 \; 0 \,].$$

The experimental results (Fig. 3) show the performance of the controller using NLP techniques to choose its parameters. It is observed that the robot reaches and remains in the desired trajectory without unwanted oscillations. In Fig. 4 it is observed that the tracking errors, when the mobile robot reaches the desired trajectory, are less than 0.06 m. However, these errors remain low compared to the mobile robot dimensions (0.508-m long, 0.497-m large, 0.277-m high). Values that the control

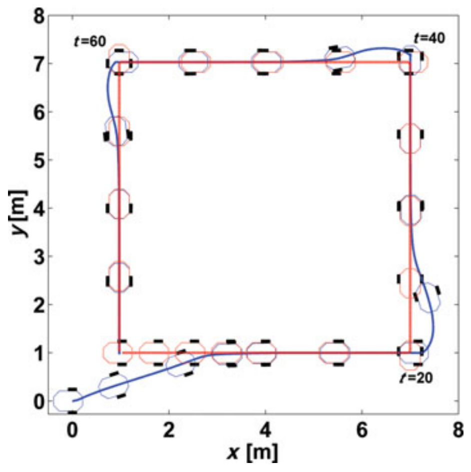

Fig. 6. Robot mobile position, experimental and desired trajectory with $T_0 = 0.1$ s. Desired trajectory: red-dot line; mobile robot position: blue line. The time is shown in some of instants together to the robots.

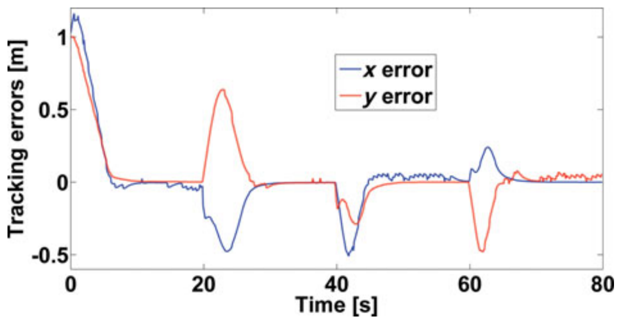

Fig. 7. Tracking errors.

actions take, so that the mobile robot follow the desired trajectory, without exceeding the maximum values of linear and angular velocities, are shown in Fig. 5(a). The values of the parameters of the controller at each sampling time are shown in Fig. 5(b). These change to ensure that the tracking errors tend to zero and the control actions do not exceed the maximum allowed. The sampling time used is $T_0 = 0.1$ s.

In different applications the trajectory to be followed by the robot is usually re-planned. This strategy can be used in applications such as obstacle avoidance and contour following. So if the danger of collision is large, the trajectory to be followed by the robot is modified abruptly and the robot must follow that path to avoid collision. Thus, the controller performance when the trajectory changes abruptly will be analyzed. In addition, in order to test the limits of our formulation, and as recommended by Roth and Batavia,[21] a square trajectory was chosen. Specifically, the robot has to track a square trajectory. The square reference trajectory is generated with constant linear velocity of $V = 0.3$ m/s. The initial position of the robot is at the system origin and the trajectory begins in the position $(x_{d(0)}, y_{d(0)}) = (1\text{ m}, 1\text{ m})$. The control action constraints considered are: $V_{\max} = 0.5$ m/s and $W_{\max} = 0.45$ rad/s.

In this case Fig. 6 shows the trajectory followed by the PIONEER 3AT mobile robot on the plane x–y, where the initial position of the mobile robot was $x = 0$ m, $y = 0$ m. It can be seen from Fig. 6 that the mobile robot tends to the desired trajectory and then follows it in a precise way. Figure 7 show how the tracking errors remain close to zero after the robot reaches the trajectory. In Fig. 8(a) is shown the time evolution of the linear and angular velocities for the mobile robot. This figure shows that the control actions do not exceed the maximum allowable. Figure 8(b) shows the controller's parameters, these parameters change to ensure that the tracking errors tend to zero and control actions do not exceed the maximum allowed.

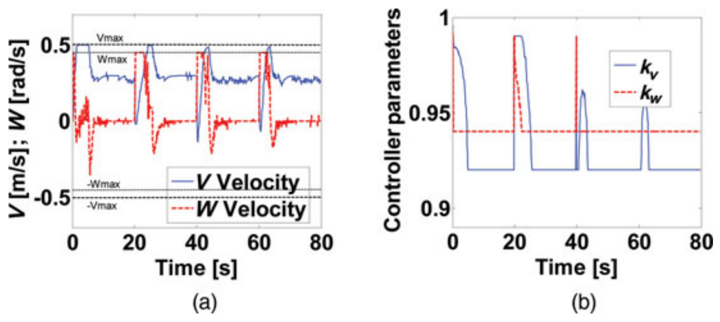

Fig. 8. (a) Control actions and the maximum allowable values. (b) Controller parameters selected using nonlinear programming techniques.

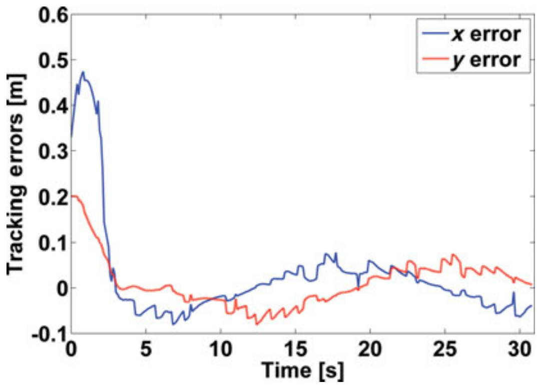

Fig. 9. Tracking errors, experimental results when the controller parameters are chosen using the techniques based on nonlinear programming. The sample time used is, $T_0 = 0.1$ s.

4.2. Experimental results with disturbances in the control actions

In this section disturbances on the control actions are added in order to test the performance of the controller, the desired trajectory is the same one used in Section 4.1. Different tests were carried out; the worst results of all are shown in this paper. The methodology proposed has a poor performance when the types of disturbances are additive and constant as shown in Eq. (26).

$$\begin{cases} \dot{x} = (V - 0.1V_d)\cos(\theta) \\ \dot{y} = (V - 0.1V_d)\sin(\theta) \,. \\ \dot{\theta} = (W - 0.1\mathrm{W}_d) \end{cases} \tag{26}$$

The experimental results obtained by the methodology in the presence of disturbance, as the ones shown in Eq. (26), are presented in Fig. 9. The sampling time used is, $T_0 = 0.1$ s.

Figure 9 shows how the tracking errors, for the proposed methodology, remain small compared to the real dimensions of the mobile robot, even in the presence of disturbances in the control actions. Figure 10 shows that the control actions do not exceed the maximum allowed values and that they remain close to $V_{d(n)}$ and $W_{d(n)}$. The tracking error can be reduced when the sampling time decreases,[1] as shown in Section 4.3.

Remark 2: The controller proposed in this paper was implemented considering disturbances in the control actions in different squared trajectories; the results can be seen in the multimedia material available at https://plus.google.com/u/0/103931265745503219057/videos.

4.3. Experimental results decreasing T_0

To enhance the controller performance the sampling time is decreased to: $T_0 = 0.05$ s. The methodology developed in this work is applied to select the controller parameters proposed by Scaglia *et al.*[1] To compare the performance of the controller, here we used the same desired trajectory as shown in Section 4.2. The disturbances in the control actions are given by Eq. (26). Figure 11

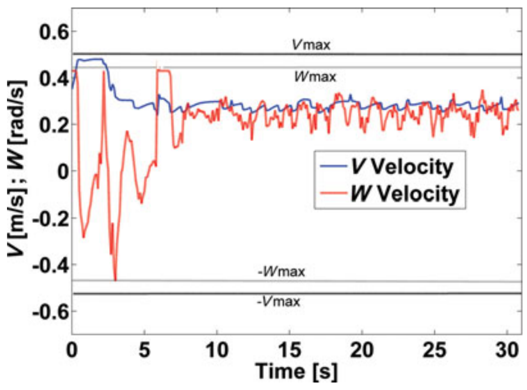

Fig. 10. Control actions when the controller parameters are chosen by techniques based on nonlinear programming.

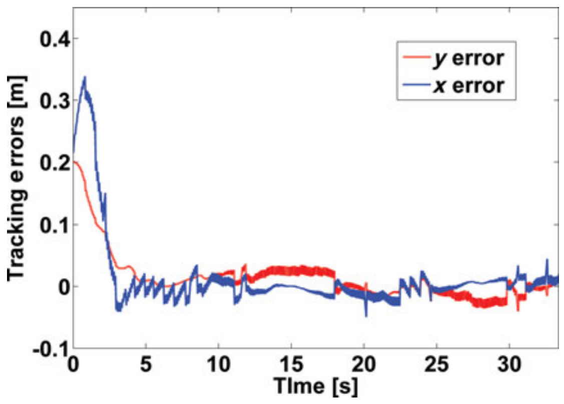

Fig. 11. Tracking errors, experimental results when the controller parameters are chosen using the techniques based on nonlinear programming. The sample time used is, $T_0 = 0.05$ s.

shows the evolution of the tracking errors. When comparing Figs. 9 and 11 it is observed that the tracking errors have considerably decreased.

In experimental results, by diminishing sample time an error reduction is observed; this reduction obeys mainly to the additive disturbance and the fact that the error from unmodeled dynamics is corrected faster, keeping errors from kinematic considerations very small.[1]

4.4. *Experimental results: comparison with the original controller*

In this section, we show how the system improves efficiency when the controller parameters are chosen by the methodology based on NLP. We choose the same trajectory as in first experiment of Section 4.1. However, with the purpose of analyzing the performance of the controller from farther away points, we choose a different initial error compared with the experiments shown in the previous sections. The initial conditions of the desired trajectory are $x_{d(0)} = 1.5$ m and $y_{d(0)} = 2$ m, and the initial conditions for the mobile robot are:

$$[\, x_{(0)} \quad y_{(0)} \quad \theta_{(0)} \,] = [\, 0 \quad 0 \quad 0 \,].$$

Two experiments are performed: first, the controller parameters are chosen by the methodology based on NLP; in the second, the controller parameters are chosen empirically in the same way as in the previous work of the authors, Scaglia *et al.*[1,6] Figures 12–14 show the results of the experiments carried out. Figure 12 shows the position of the robot in the plane (x, y). As can be seen in Fig. 12, both controllers have similar performances. Nevertheless, Fig. 13 shows that the control actions obtained by the controller proposed in this work never exceed the maximum value allowed.

Figure 13 shows the values take by the control actions in the experiences. Figure 13(a) shows that the values of control actions, when the controller parameters are chosen empirically, saturate

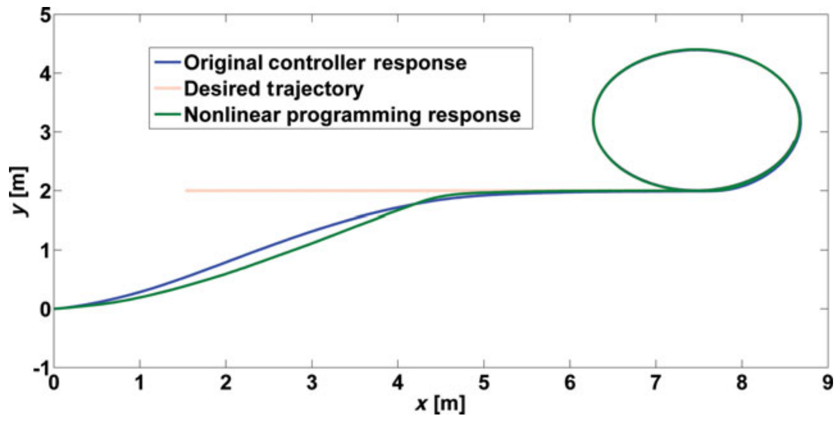

Fig. 12. Robot mobile position, experimental and desired trajectory with $T_0 = 0.1$ s. Desired trajectory: red-dot line; mobile robot position when the controller parameters are chosen by empirical tests: blue line. Mobile robot position when the controller parameters are selected using nonlinear programming techniques: green line.

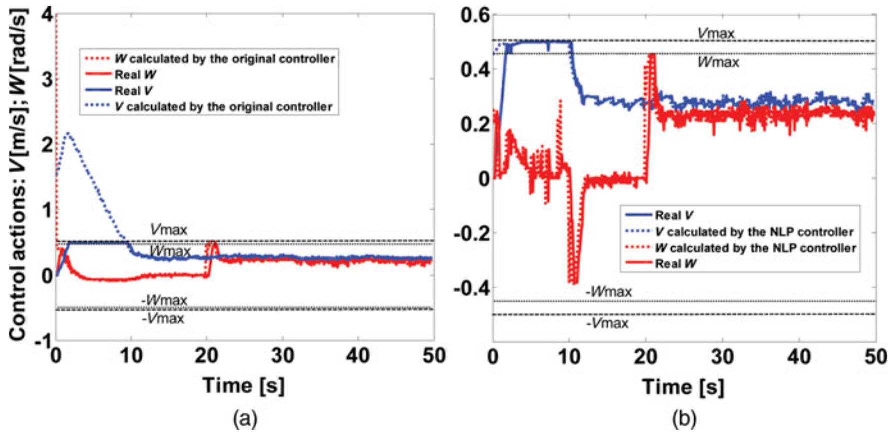

Fig. 13. (a) Control actions response, when the controller parameters are chosen by empirical tests. (b) Control actions response, when the controller parameters are selected using nonlinear programming techniques.

the actuators at the beginning. In contrast, Fig. 13(b) shows that this behavior is avoided when the controller parameters are chosen through NLP. Figure 14 shows such controller parameters.

In the original controller presented by the authors in refs. [1] and [6], the parameters were chosen by empirical tests. Then, when the tracking error is too big, the control actions might exceed their maximum and minimum values, producing the saturation of the actuators. Thus, it loses the feedback loop of the system as shown in Fig. 13(a). A further advantage of the methodology proposed in this paper is that when the trajectory changes, there is no need to change the minimum limit of the parameters, since the values are chosen by optimization. However, when the parameters are chosen empirically it is convenient to find the best fitting parameters for a better performance of the controller according to the chosen trajectory.

A way to compare the performances obtained between the controller proposed in refs. [1] and [6] and the controller proposed here, is by using a cost function. As previously presented in ref. [9], let Φ be a desired trajectory, where $\# \Phi$ is the number of points of such trajectory. In this paper we propose a cost function based on the traction control effort and the heading control effort:

$$C^{\Phi} = C_v^{\Phi} + C_w^{\Phi} = \sum_{i=0}^{\#\Phi} \frac{1}{2} V_{(i)}^2 + \sum_{i=0}^{\#\Phi} \frac{1}{2} W_{(i)}^2.$$

Table I. Summary of the costs obtained by the controllers.

Cost function/Controller	C1	C2
$C^{\Phi} = C_v^{\Phi} + C_w^{\Phi} = \sum_{i=0}^{\#\Phi} \frac{1}{2} V_{(i)}^2 + \sum_{i=0}^{\#\Phi} \frac{1}{2} W_{(i)}^2$	$C^{\Phi} = 72.165$	$C^{\Phi} = 61.583$

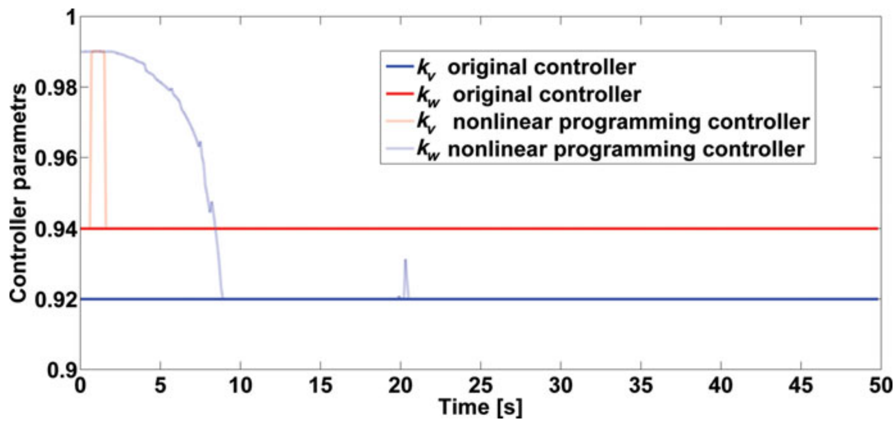

Fig. 14. Controller parameters.

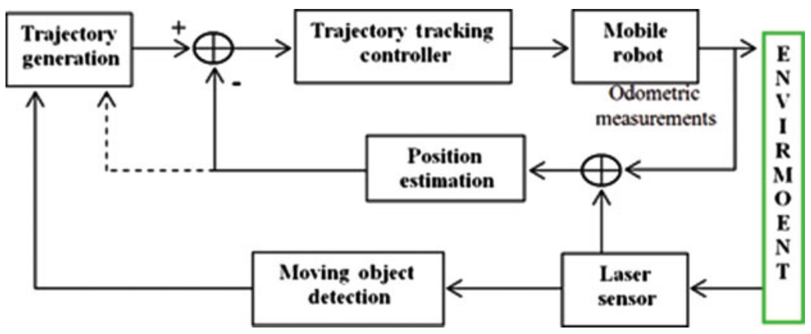

Fig. 15. System architecture of the reaching a moving target case.

The expressions that define the traction effort and the heading effort are defined in ref. [9]. The cost obtained by the proposed controller in refs. [1] and [6] will be called C1 and the cost obtained by the controller proposed in this paper will be called C2.

The costs obtained for each controller are shown in Table I. By observation, the cost obtained by the controller proposed herein (C2) is less than the cost obtained by the original controller. By inspection, it can be seen in Fig. 12 that in both controllers the tracking errors tend to zero, however the effort required by our controller to meet the target is 20% less.

It is important to remark that the controller developed in refs. [1] and [6] does not ensure the convergence to zero of the tracking errors when the control action computed by the controller exceeds the limits of saturation. In this situation the proof of convergence to zero of the tracking errors is invalidated, because the robot is not capable of responding to commands from the controller. This is one of the main advantages of the controller proposed in this paper with respect to the controller proposed in refs. [1] and [6].

4.5. Reaching a moving target: a case study

The trajectory-tracking controller shown in this work is also implemented in a *reaching a moving target* case. The aim of this case study is to position the robot within a neighborhood of a moving agent and, once positioned, the mobile robot is able to follow such an agent. This section focuses on the *reaching* problem only. Figure 15 shows the architecture of the strategy presented herein, which can be summarized as follows:

- The mobile robot and the moving agent are located within an indoor environment. In this work, we have used a person as a mobile agent.
- A range laser scanner implemented in the mobile robot acquires range information from the environment. By comparing successive scanner measurements, the system is able to detect the moving agent within the environment, as shown in ref. [11]. If more than one moving agent is present, then a matching stage in Fig. 15 would aid in the agents recognition process. In this work, and for experimental purposes, the subject was the only moving agent present in the environment (despite the mobile robot).
- A positioning system, based on SLAM (Simultaneous Localization and Mapping) algorithm is also implemented on the mobile robot. The SLAM algorithm used was previously published by the authors of refs. [12] and [13]. It extracts corners and walls from laser range measurements to construct a geometric map of the environment while localizing the vehicle within it. In addition, the SLAM algorithm is implemented in an Extended Kalman Filter (EKF). Further details of the EKF–SLAM implementation and consistency tests can be found in refs. [12] and [13]. Such a SLAM-based positioning system uses the range laser measurements, the motion control commands, and the odometric measurements to estimate both the map of the environment and the position of the robot within it, minimizing estimation errors.
- The estimated position of the robot and the position of the detected moving agent (both referenced to a global Cartesian system) are used by the trajectory planning stage to generate a feasible path between the robot and a neighborhood of the moving agent. The trajectory planning stage used in this work is a Monte-Carlo-based trajectory generation that finds an optimum and safe (collision-free) trajectory between the robot and a previously defined goal. Such a trajectory is a circled-shape trajectory and it is kinematically compatible with both unicycle and car-like type of mobile robots. Details of the trajectory planning strategy used in this work can be found in a previous work of the authors of ref. [14].
- The trajectory planning strategy generates a reference trajectory that is compared to the current estimated pose of the mobile robot. The comparison results are fed to the trajectory-tracking controller proposed in this work.
- The trajectory planning strategy successively replans the trajectory at each sampling time (i.e., it is a dynamic trajectory planning[15]). The latter is necessary due to the fact that the moving agent changes its position in time within the environment. In addition, the moving agent follows a random path.
- The system stops its execution once the robot is positioned in a neighborhood of the moving agent. Then, other approaches can be used to follow the moving agent, as previously stated.

The experimentation shown in this section was carried out under the following considerations: the maximum velocities of the robot were set to $|V_{\max}| = 1\text{m/s}$ and $|W_{\max}| = 2.5\text{rad/s}$.

It is expected for the moving agent's velocity to be smaller than the maximum velocities of the robot (otherwise, the robot will not reach a neighborhood of the moving agent). The sampling time of the system was set to $T_0 = 0.2$ s. The initial position of the robot was set to $[\, x_{(0)} \;\; y_{(0)} \;\; \theta_{(0)}\,] = [\,-9 \;\; 2 \;\; 0\,]$. In addition, once the robot enters into a neighborhood of radius 1 m of the moving agent, the system stops its execution. Although we have used *radious* = meter, other criteria can be used instead. The mobile robot used is Pioneer 3AT previously mentioned in Section 4.

Figure 16 shows three experimental cases of the system shown in Fig. 15 in a free-collision environment. In Figs. 16(a), 16(c), and 16(e), the solid red dots are raw laser data; solid yellow squares represent the robot position at each sampling time. In this case, and considering the absence of objects within the scanned workspace, the robot's localization process is performed only by the odometric measurements of the vehicle. The trajectory followed by the robot was planned by the trajectory planning stage in Fig. 15. The blue circles represent the position, within the environment, where the moving agent was detected. The solid grey area represents the workspace sensed by the range laser scanner. As can be seen, the robot successfully reaches a neighborhood of the moving agent. Figures 16(b), 16(d), and 16(f) show the control command signals for Figs. 16(a), 16(c), and 16(e), respectively. The solid blue line represents the linear velocity command, whereas the solid green line represents the angular velocity. As can be seen, the control command signals remain bounded by their maximum values. In addition, Fig. 17 shows the proposed reaching strategy within a confined space. The solid black lines are associated with walls and line-shaped objects from the environment.

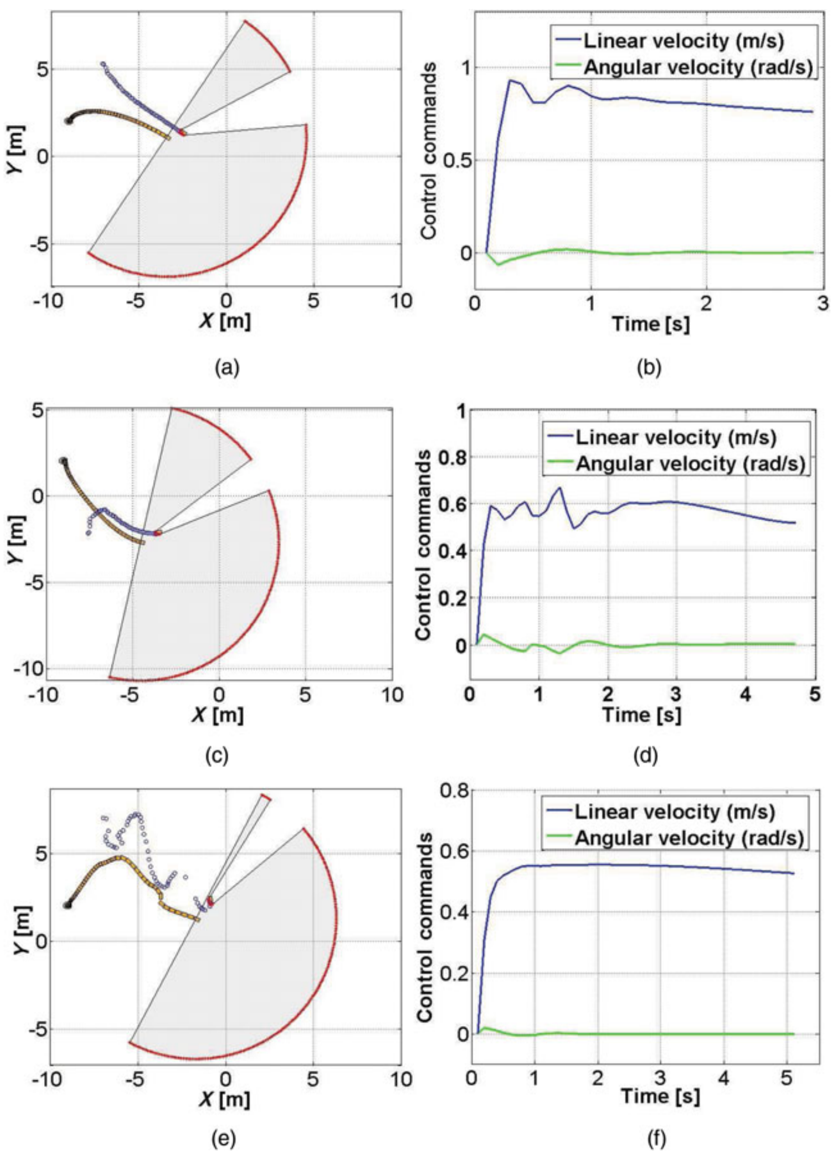

Fig. 16. Collision-free environment cases. (a), (c), and (e) show three examples of the proposed reaching strategy, whereas (b), (d), and (f) show their respective control command signals.

For reconstruction of the map shown in Fig. 17, we have used the EKF–SLAM algorithm previously mentioned in ref. [13]. The solid grey squares represent the path traveled by the mobile robot and estimated by the SLAM algorithm; the blue circles represent the location of the moving object at the moment of its detection. As can be seen, the mobile robot reaches a neighborhood of the moving agent. Figures 17(a) and 17(c) show two different experiments within different environments; whereas Figs. 17(b) and 17(d) show their respective control command signals.

As can be seen, the control command signals remain bounded by their maximum values. It is worth mentioning that the missing blue circles in Fig. 17(a) are due to miss detections of the moving agent. Furthermore, Figs. 16 and 17 show that, as the robot approaches the moving agent, the planning strategy makes the controller to increase the linear velocity of the vehicle. The latter is a planning issue due to the fact that, the planning stage always searches for a path that allows for the robot to reach the moving agent in the minimum time (thus increasing its velocity).

Figure 18 shows how is the evolution of the distance between the mobile robot and the moving agent in time, for several experiments carried out in this work. As can be seen, the distance among them is gradually reduced (i.e., the robot reaches the neighborhood of the moving agent).

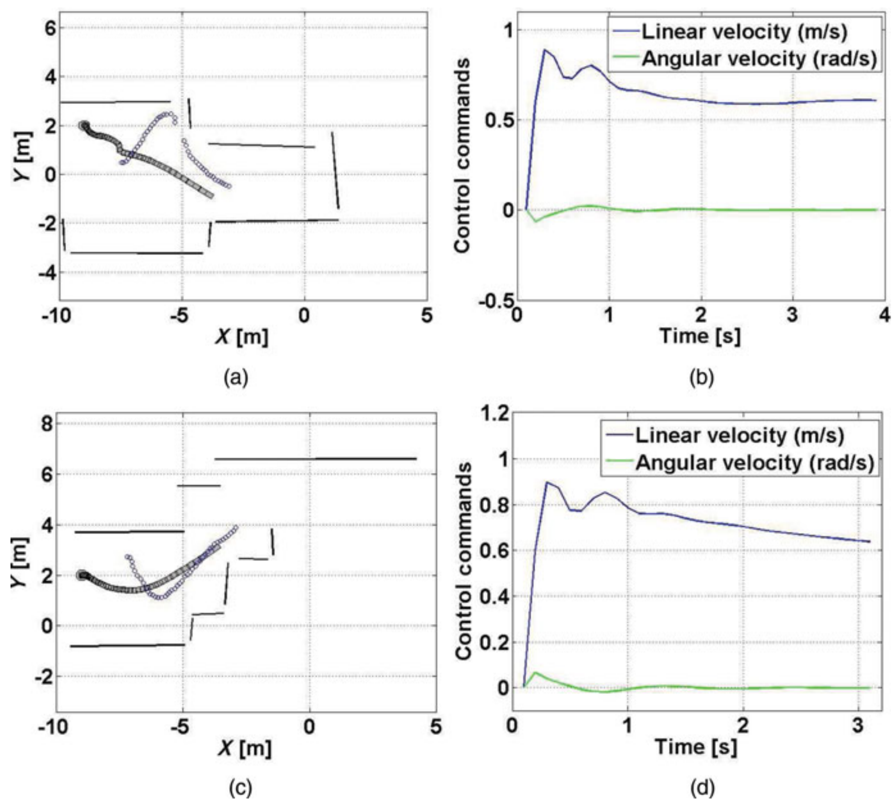

Fig. 17. SLAM-based case. (a) and (c) show two examples of the proposed reaching strategy within an environment mapped by the EKF–SLAM algorithm, whereas (b) and (d) show their respective control command signals.

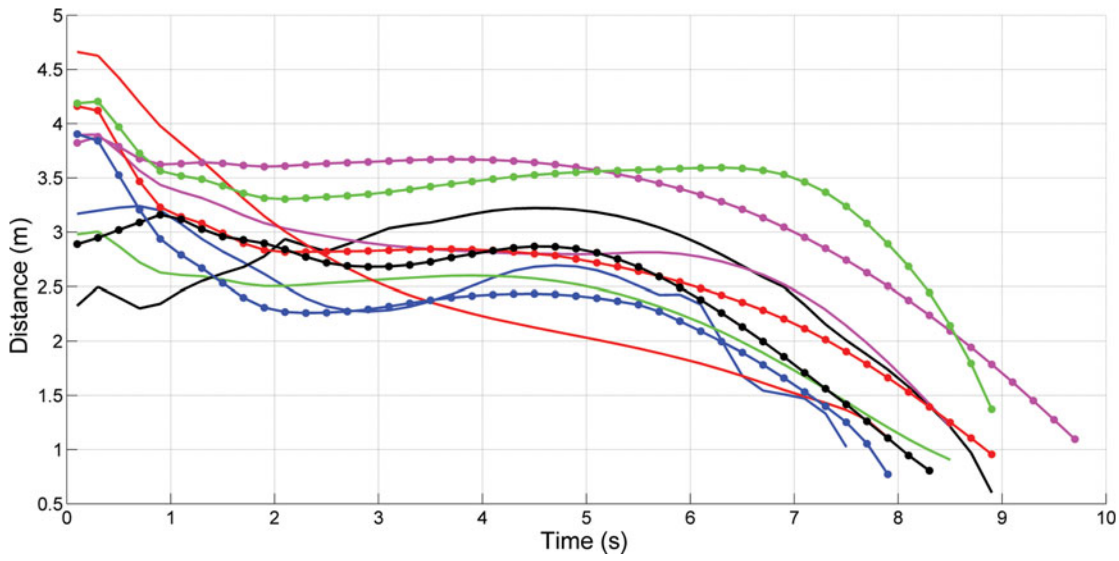

Fig. 18. Evolution, in time, of the distance between the mobile robot and the moving agent for several experiments.

5. Discussions

– The optimization techniques introduced in Section 3.2 result in an optimal constant value on each sampling period to ensure a faster convergence avoiding the actuator's saturation. The methodology based on NLP directly considers the design constraints imposed.

– In our study, the designed controller does not present the disadvantage of the controller proposed by Sun and Cui,[4] where a linear velocity different from zero or only certain special cases are necessary. Furthermore, our controller does not need to change the control expression when the angular velocity is lower than a pre-established value as in ref. [4].
– As stated in Section 4.5, as the robot approaches the moving agent, the controller increases the vehicle's linear velocity (in order to reach a neighborhood of the moving agent as soon as possible). The latter can be dangerous for both the vehicle and the moving agent, due to the fact that as the velocity increases, the acceleration increases and the vehicle will need more space to put a brake. Thus, such a neighborhood of reaching must be properly defined before navigation.
– This is a continuation work of the authors. In the previous work, the authors did not consider the restrictions on control actions; in this paper such restrictions were included in the controller design. The latter allows reach the goal without exceeding the maximum control actions permissible.

6. Conclusions

A new method to find the controller parameters that allow the tracking trajectory of a mobile robot subject to saturation in the control actions has been presented. This novel technique allows finding the values of the control actions (that cause the tracking errors to tend to zero) without exceeding the maximum allowable values. One of the most significant contributions of this work involves the application of a method to find the parameters of the controller developed by the authors in refs. [1] and [6]. This method can be used to find values of the control actions that do not exceed the actuators saturation limits.

Different tests were carried out to demonstrate the effectiveness of the proposed methodologies. The experimental results show that the laboratory tracking errors obtained can be reduced by lowering the sampling time. Real world experiments (Section 4.5) show an example of an application in which the robot follows a moving target. In addition, it was shown that when the methodology proposed here was compared with the original controller, the control actions did not saturate the actuators. The values found by our approach maintain control actions below the saturation values while the tracking errors tend to zero. The experimental results demonstrated the effectiveness of the technique and verified the theoretical results.

Acknowledgements

The authors would like to thank Universidad Nacional de San Juan, CONICET (Argentina) and Universidad Técnica Federico Santa María Valparaíso (Chile) for partially funding this research.

References

1. G. J. E. Scaglia, L. M. Quintero, V. Mut and F. Di Sciascio, "Numerical methods based controller design for mobile robots," *Robotica* **27**(2), 269–279 (2009).
2. T. C. Lee, K. T. Song, C. H. Lee and C. C. Teng, "Tracking control of unicycle modeled mobile robots using a saturation feedback controller," *IEEE Trans. Control Syst. Technol.* **9**(2), 305–318 (2001).
3. G. Klancar and I. Skrjanc, "Tracking-error model-based predictive control for mobile robots in real time," *Robot. Auton. Syst.* **55**(6), 460–469 (2007).
4. S. Sun and P. Cui, "Path tracking and a practical point stabilization of mobile robot," *Robot. Comput.-Integr. Manuf.* **20**, 29–34 (2004).
5. G. Oriolo, A. Luca and M. Vandittelli, "WMR control via dynamic feedback linearization: Design, implementation, and experimental validation," *IEEE Trans. Control Syst. Technol.* **10**(6), 835–852 (2002).
6. G. J. E. Scaglia, A. Rosales, L. M. Quintero, V. Mut and R. Agarwal, "A linear-interpolation-based controller design for trajectory tracking of mobile robots," *Control Eng. Practice* **18**, 318–329 (2010).
7. A. Rosales, G. J. E. Scaglia, V. Mut and F. Di Sciascio, "Formation control and trajectory tracking of mobile robotic systems – A linear algebra approach," *Robotica* **29**(3), 335–349 (2011).
8. A. Rosales, G. J. E. Scaglia, V. Mut and F. Di Sciascio, "Trajectory tracking of mobile robots in dynamic environments – A linear algebra approach", *Robotica* **27**, 981–997 (2009).
9. F. A. Cheein and G. J. E. Scaglia, "Trajectory tracking controller design for unmanned vehicles: A new methodology," *J. Field Robot.* doi:10.1002/rob.21492, (2013). http://onlinelibrary.wiley.com/doi/10.1002/rob.21492/abstract.
10. J. N. Rico, I. Alcala, J. Gomez-Ortega and E. Camacho, "Mobile robot path tracking using PID controller," *Control Eng. Practice* **9**, 1209–1214 (2001).

11. P. Biber and T. Duckett, "Dynamic Maps for Long-Term Operation of Mobile Service Robots," *Proceedings of Robotics: Science and Systems*, MIT Press, Cambridge, MA (Jun. 8–11, 2005) pp. 17–24.
12. F. A. Auat Cheein and R. Carelli, "Analysis of different feature selection criteria based on a covariance convergence perspective for a SLAM algorithm," *Sensors (Basel)* **11**(1) 62–89 (2011).
13. F. A. Auat Cheein, F. di Sciascio, G. J. E. Scaglia and R. Carelli, "Towards features updating selection based on the covariance matrix of the SLAM system state," *Robotica (Cambridge)* **29**(2), 271–282 (2010).
14. F. Auat Cheein, C. De la Cruz, R. Carelli and T. Bastos, "Navegacion Autonoma Asistida basada en SLAM para una Silla de Ruedas Robotizada en Entornos Restringidos," *Revista Ibero Americana de Automatica e Informatica Industrial* **8**(2), 81–92 (2011).
15. W. Dong-Shu and Y. Hua-Fang, "Path Planning of Mobile Robot in Dynamic Environments," *Proceedings of the International Conference on Intelligent Control and Information Processing (ICICIP)*, (Jul. 25–28, 2011) pp. 691–696.
16. R. Agarwal, *Difference Equations and Inequalities, Theory, Methods, and Applications* (Marcel Dekker, Inc., New York, 2000).
17. E. O. Omojokun, Trust Region Algorithms for Optimization with Nonlinear Equality and Inequality Constraints *Ph.D. Thesis* (Boulder, CO: University of Colorado Boulder, 1989).
18. M. S. Bazaraa, H. D. Sherali and C. M. Shetty, *Nonlinear Programming Theory and Algorithms*, (John Wiley & Sons, Inc., Hoboken, NJ, 2006) 872 pp.
19. S. Jung, P. Jeon and T. C. Hsia, "Contour tracking of an unknown planar object by regulating force for mobile robot navigation," *Robotica* **25**, 297–305 (2007).
20. J. M. Toibero, F. Roberti and R. Carelli, "Stable contour-following control of wheeled mobile robots," *Robotica* **27**, 1–12 (2009).
21. S. Roth and P. Batavia, "Evaluating path tracker performance for outdoor mobile robots," *Automation Technology for Off-Road Equipment*, July, 2002. Retrieved on December 4, 2012, from http://www.ri.cmu.edu/publication_view.html?pub_id=3908.
22. J. N. Rico, J. Gomez-Ortega and E. Camacho, "A Smith predictor based generalized predictive controller for mobile robot path-tracking," *Control Eng. Pract.* **7**, 729–740 (1999).