The Null-Space based Behavioral control for non-holonomic mobile robots with actuators velocity saturation

F. Arrichiello, S. Chiaverini, P. Pedone, A. A. Zizzari and G. Indiveri

Abstract— This paper presents the application of the Null-Space based Behavioral (NSB) approach to the motion control of a non-holonomic mobile robot with velocity saturated actuators. In particular, the proposed solution aims at managing actuator velocity saturations by dynamically scaling task velocity commands so that the hierarchy of task priorities is preserved in spite of actuator velocity saturations. The approach is tested on a specific case study where the NSB approach elaborates the motion directives for a mobile robot that has to reach a target while avoiding a punctual obstacle. The approach is validated by numerical simulations and by experimental results with a non-holonomic mobile robot.

I. INTRODUCTION

Mobile robots have been object of widespread research in the last decades. Their applications span over service, industrial, military and civil fields and involve missions like exploration, transportation and mobile manipulation. In spite of the many advancements in the field of mobile robotics, several challenging issues are still open.

With reference to motion control problems, widely different methods and techniques have been presented in the literature including behavior-based approaches. These appear to be appealing because they give the system the autonomy to navigate in complex environments avoiding (or at least limiting) the need of off-line path planning by using sensors to obtain instantaneous information of the environment with an increased overall flexibility. Indeed, behavior-based approaches have been shown to be rather suitable to navigate in unknown or dynamically changing environments. Among the behavioral approaches, seminal works are reported in the papers [11] and [8], while a comprehensive state of the art is presented in [9]. Behavioral approaches have been also applied to the formation control of multi-robot systems as in, e.g., [17], [15] and [10].

Extending the idea of inverse kinematics techniques for industrial manipulators to the case of mobile robots, a new behavior based approach, namely the Null-Space-based Behavioral control (NSB), has been recently presented in the literature [4]. In particular, the NSB approach is based on the idea of task based kinematical control for industrial manipulators presented in [14], [16], [18] of exploiting eventual kinematical redundancy to try to simultaneously accomplish more than one motion tasks. Nevertheless, as discussed in [12], just in the case of conflicting tasks it is necessary to devise singularity-robust algorithms that ensure proper functioning of the inverse velocity mapping. Based on these works, this idea is developed in [6] in the framework of the singularity-robust task-priority inverse kinematics [12]. In [7] this approach has been used in simulation to control a multirobot team performing a caging mission, subject to obstacle avoidance and failure of one or more vehicles. The NSB has been introduced in comparison with the main behavior-based approaches in [4] and it has been recently experimentally applied in a large number of multi-robot missions such as formation control or escort/entrapping an autonomous target [3].

Ever since the idea of task based kinematics control has been developed, main stream research has focused in particular on extending the approach to the case of many concurrent tasks, on the definition of useful lower priority tasks and on coping with mathematical singularities. Minor attention has been instead devoted to the problems that arise in the presence of actuator torque or velocity saturations. Within a purely kinematics framework, should a low priority task command induce even a single actuator to saturate its velocity, this could irreversibly corrupt the high priority task: on the other hand, the common work-around to scale all the actuator speeds to avoid saturations when the original command is too high has the drawback that so doing higher priority tasks are slowed down due to lower priority ones hence somehow violating the very basic idea of priority hierarchy. To avoid these limitations, in this paper the standard NSB approach is extended by the saturation management technique presented in [13]. The proposed solution is tested, both in simulation and experimentally, on the specific case of the motion control of a single mobile robot with velocity saturated actuators that has to reach a target while avoiding a punctual obstacle. The adopted experimental platform is a Khepera II mobile robot.

II. THE NULL SPACE BASED BEHAVIORAL CONTROL

Following the main behavior-based approaches and similarly to task-based kinematic control approaches, the overall mission for the autonomous robot is decomposed in elementary sub-problems (or tasks) that have to be simultaneously managed. For each task a suitable function is mathematically defined and a motion control directive is elaborated. Then, the single tasks that compose the overall mission are organized in a proper priority order. Finally, the global
motion control directive to the robot is elaborated by suitably combining the motion directives of the single tasks. In details, by defining as $\sigma \in \mathbb{R}^n$ the task variable to be controlled and as $p \in \mathbb{R}^n$ the system configuration, then:

$$\sigma = f(p)$$  \hspace{1cm} (1)

with the corresponding differential relationship:

$$\dot{\sigma} = \frac{\partial f(p)}{\partial p} \dot{v} = J(p)v,$$

(2)

where $m$ is the task function dimension, $J \in \mathbb{R}^{m \times n}$ is the configuration-dependent task Jacobian matrix and $v \in \mathbb{R}^n$ is the system velocity. Notice that $n = 2$ in case of a material point moving on a planar surface.

An effective way to generate motion references for the vehicles starting from desired values $\sigma_d(t)$ of the task function is to act at the differential level by inverting the (locally linear) mapping (2); in fact, this problem has been widely studied in robotics (see, e.g., [19] for a tutorial). A typical requirement is to pursue minimum-norm velocity, leading to the least-squares solution:

$$v_d = J^t (\sigma_d + \Lambda \sigma),$$  \hspace{1cm} (3)

where $J^t = J^T (JJ^T)^{-1}$ (when $J(q)$ is full rank). $\Lambda$ is a suitable constant positive-definite matrix of gains and $\sigma$ is the task error defined as $\sigma = \sigma_d - \sigma$. It is worth noticing that the term $\Lambda \sigma$ is added to counteract the numerical drift due to discrete-time integration.

When the mission is composed by multiple tasks the overall vehicle velocity is elaborated by properly merging the outputs of the single tasks. In particular, each task velocity is computed as if it were acting alone; then, before adding its contribution to the overall vehicle velocity, a lower-priority task is projected onto the null space of the immediately higher-priority task so as to remove those velocity components that would conflict with it. Thus, on the analogy of eq. (3), the single task velocity is computed as

$$v_i = J^t_i (\sigma_{i,d} + \Lambda_i \sigma_i),$$  \hspace{1cm} (4)

where the subscript $i$ denotes $i$-th task quantities and the priority of the task. In the case of 3 tasks according to [12] the solution (3) is modified into

$$v_d = v_1 + (I - J_1^T J_1) [ v_2 + (I - J_2^T J_2) v_3 ],$$  \hspace{1cm} (5)

where the $(I - J_i^T J_i)$ operator represents the null-space projector of the $i$-task. It is worth noticing that to guarantee the mission stability the tasks have to verify the properties defined in [5].

III. THE SATURATION MANAGEMENT TECHNIQUE

The above described procedure guarantees the compatibility among tasks in the assumption that the overall velocity given by equation (5) does not exceed in norm the maximum value, say $v_{\text{max}}$, that is physically realizable. Indeed hardware and energy limitations imply that the maximum possible velocity should be bounded. If the commanded velocity $v_d$, due to the presence of lower priority tasks, should be such that $\|v_d\| > v_{\text{max}}$, the actual speed of the vehicle relative to a $v_d$ command would result in $v_{\text{max}} \|v_d\|$; in such case, in spite of the null-space projection technique, lower priority tasks would actually still conflict with higher priority ones at least in terms of task error convergence rate.

To overcome this issue, task velocity commands should be normalized in norm so that lower priority commands do not conflict with higher priority ones due to saturation effects. Consider the function $s : \mathbb{R} \times [0, \infty) \rightarrow \mathbb{R}$ such that

$$s(x, c) = \begin{cases} 0 & \text{if } x = 0 \\ \frac{1}{c} & \text{if } 0 < |x| < c \\ c/|x| & \text{otherwise} \end{cases}$$  \hspace{1cm} (6)

where the non negative second argument $c$ of $s(x, c)$ will be called the capacity of $x$. By definition $s(x, c)$ is simply a non-negative scalar scaling factor such that $x s(x, c)$ is "clipped" to $c \text{sign}(x)$ whenever $|x|$ should exceed the capacity $c$ and is equal to $x$ otherwise, i.e. $x s(x, c)$ is simply the saturated version of $x$ in the range $[-c, c]$. Also notice that by its very definition

$$s(x, 0) = 0 \ \forall \ x,$$

namely if $x$ should be assigned zero capacity, then $x s(x, 0) = 0$ for any value of $x$. Consider then the generalization of equation (5) written in the form

$$v_d = \sum_{i=1}^N \tilde{v}_i$$  \hspace{1cm} (8)

being

$$\tilde{v}_1 = v_1$$  \hspace{1cm} (9)

$$\tilde{v}_2 = (I - J_1^T J_1) v_2$$  \hspace{1cm} (10)

$$\tilde{v}_3 = (I - J_2^T J_2) (I - J_3^T J_3) v_3$$  \hspace{1cm} (11)

In order to take into account the upper bound on the norm of the vehicle’s velocity, equation (8) is modified as follows:

$$v_d = \sum_{j=1}^N \tilde{v}_j s(\|\tilde{v}_j\|, c_j)$$  \hspace{1cm} (12)

where each task capacity is recursively and dynamically computed as:

$$c_1(t) \leq v_{\text{max}} \ \text{(constant, i.e. } \dot{c}_1(t) = 0)$$

$$c_2(t) = c_1 - \|\tilde{v}_1\| s(\|\tilde{v}_1\|, c_1)$$

$$c_3(t) = c_2(t) - \|\tilde{v}_2\| s(\|\tilde{v}_2\|, c_2(t))$$

$$\vdots$$

$$c_N(t) = c_{N-1}(t) - \|\tilde{v}_{N-1}\| s(\|\tilde{v}_{N-1}\|, c_{N-1}(t)).$$  \hspace{1cm} (13)

Notice that by construction all the above task capacities are non negative, i.e. $c_j \geq 0 \ \forall \ j \in [1, N]$, and that

$$c_j \leq c_{j-1} \ \forall \ j \in [2, N].$$
c_i = 0 \implies c_j = 0 \quad \forall \ j > i

namely if a given task is assigned zero capacity, all the lower priority tasks will also automatically have zero capacity and all their weights in the sum (12) will be zero. The capacity of task $i$ can be viewed as the residual capacity after the higher priority task $i - 1$ has been commanded; thus, by example, $c_2$ will be zero (and also $c_j : j > 2$) if the task 1 input $v_1$ is saturating all its capacity $c_1$. In words, each task will be commanded with a non null weight only if the higher priority task have not saturated. The fact that $c_1$ needs not to exceed $v_{\text{max}}$ is due to the fact that task 1 alone should not saturate the actuator capacity $v_{\text{max}}$; moreover, given that $c_{j+1} \leq c_j \forall \ j \in [1, n - 1]$, the condition $c_1 \leq v_{\text{max}}$ guarantees that each term in the sum equation (12) will have Euclidean norm smaller or equal to the threshold $v_{\text{max}}$. Most important, following the same kind of proof reported in [13], it can be shown that also $v_d$ in equation (12) has Euclidean norm smaller or equal than $v_{\text{max}}$.

IV. CASE STUDY

In order to test the conjuncted action of the NSB approach with the saturation management technique, in this section we will analyze the case study of a single non-holonomic mobile robot that has to perform an autonomous mission. After the definition of the proper task functions for the robot-robot kinematic model, simulation and experimental results will be discussed in order to validate the proposed approach.

A. Mission definition: move-to-goal and obstacle-avoidance

The robot mission is decomposed in two elementary tasks:

- Task #1: obstacle-avoidance;
- Task #2: move-to-goal.

a) Obstacle-avoidance

Obstacle-avoidance is the highest priority task because its achievement is of crucial importance to preserve the integrity of the vehicle. In presence of an obstacle in the advancing direction, the task aim is to keep the robot at a safe distance from the obstacle. Thus, its implementation elaborates as output a velocity, in the robot-obstacle direction, that keeps the robot to a safe distance from the obstacle. Therefore, it is:

$$\sigma_1 = \|p - p_o\| \in \mathbb{R}; \quad \sigma_{1,d} = d; \quad J_1 = \hat{r}^T \in \mathbb{R}^{1 \times 2},$$

where $p_o$ is the obstacle position, $d$ is the safety distance from the obstacle and

$$\hat{r} = \frac{p - p_o}{\|p - p_o\|}$$

is the unit vector aligned with the obstacle-to-vehicle direction. According to (4), the primary-task velocity is then

$$v_1 = J_1^T \lambda_1 (d - \|p - p_o\|).$$

The expression of the corresponding null-space is

$$N(J_1) = I - J_1^T J_1 = I - \hat{r} \hat{r}^T$$

where $I$ is the Identity matrix of proper dimensions. It can be observed that $v_1$ is a vector in the obstacle-to-vehicle direction while the null-space operator does project in the direction tangent to the circle centered in $p_o$ and passing through $p$.

b) Move-to-goal

The move-to-goal task output is a velocity, in the vehicle-to-goal direction, proportional to the distance from the goal $p_g$; therefore, it is

$$\sigma_2 = p \in \mathbb{R}^2; \quad \sigma_{2,d} = p_g; \quad J_2 = I \in \mathbb{R}^{2 \times 2}.$$ 

According to (4), the secondary-task velocity then is

$$v_2 = A_2 (p_g - p).$$

It is worth noticing that the obstacle avoidance task is active only when required, i.e., when the vehicle is closer than a threshold value to the obstacle and when the output velocity of the lower priority tasks is in the obstacle direction. Thus, when the obstacle avoidance is deactivated, its output is null while its null space projector is the identity matrix ($\hat{v}_1 = 0$, $\hat{v}_2 = v_2$). Otherwise, when the activations conditions are both simultaneously verified, the tasks are activated and their outputs are combined as previously described ($\hat{v}_1 = v_1$, $\hat{v}_2 = (I - J_1^T J_1)v_2$).

B. Unicycle-like kinematic model

![Fig. 1. Khepera II mobile robot: unicycle-like mobile robot manufactured by K-Team and its kinematics model](image)

Assume that the vehicle to be controlled is a differential drive wheeled mobile robot moving in a horizontal plane. In order to implement on such a mobile robot the kinematics control solution described in section II, the systems’ velocity $v_d$ should be assigned at each time instant. Since the NSB outputs a linear velocity for a material point, while the mobile robot actuators are its wheels, it is necessary to convert the NSB output $v_d$ to wheels’ desired velocities $\omega$.

With reference to Figure 1 and the variables there depicted, the mapping between wheels’ angular velocities $^1(\omega_l, \omega_r)^T$ and the velocity of point $C$ (wheels axis center) is given by:

$$\begin{pmatrix} v_C \\ \omega \end{pmatrix} = \frac{r}{2} \begin{pmatrix} 1 & 1 \\ -\frac{1}{b} & \frac{1}{b} \end{pmatrix} \begin{pmatrix} \omega_l \\ \omega_r \end{pmatrix}$$

where $^1\omega_l$, $^1\omega_r$ are the left and right wheels’ angular velocity vectors respectively.
being $v_C \mathbf{i}_1$ and $\omega (\mathbf{i}_1 \times \mathbf{j}_1)$ the linear and angular velocity vectors of point C (notice also that $\omega = \theta$).

Yet given the non-holonomic nature of the mobile robot, the velocity of point C cannot be arbitrarily assigned as it must be always aligned with $\mathbf{i}_1$. To overcome this difficulty, the proposed control solution can be applied to a point P as depicted in Fig. 1. Denoting with $v_P = (v_P(x), v_P(y))^T$ the velocity of point P in the absolute frame $(i_0, j_0)$, standard kinematics relations allow to express wheels’ speeds in terms of $v_P$ as:

$$
\begin{bmatrix}
\omega_l \\
\omega_r
\end{bmatrix} = \frac{1}{r} \begin{bmatrix}
1 & -\frac{b}{r} \\
1 & \frac{b}{r}
\end{bmatrix} \begin{bmatrix}
c_0 & s_0 \\
s_0 & c_0
\end{bmatrix} \begin{bmatrix}
v_P(x) \\
v_P(y)
\end{bmatrix},
$$

(15)

being $c_0 := \cos \theta$ and $s_0 := \sin \theta$. As for the choice of the parameter $\Delta_P$, this should be small enough to let point $P$ stay in the footprint of the vehicle. The specific choice $\Delta_P = b$ guarantees isotropy in the sense that the norm of $(\omega_l, \omega_r)^T$ does not depend on the orientation of $v_P$.

C. Where introducing the saturations?

As reported in Section III, the velocity commands $\bar{v}_j$ relative to each task $j$ can be normalized by the functions $s(\|v_j\|, c_j)$: the resulting total command $v_d$ given by equation (12) is then mapped into joint (i.e. wheels) speeds according to the mod el described in Section IV-B.

An alternative procedure to guarantee bounded velocity commands is to first map the $v_j$ terms on wheels’ speeds $\omega_j$ through equation (15), and then to normalize the resulting wheels’ speed commands in the form $\omega_j s(\|\omega_j\|_\infty, c_j)$. A detailed analysis of the differences relative to these two methods goes beyond the scope of this paper and will not be discussed here for the sake of brevity. Nevertheless, given that the mapping between vehicle and joint speeds is bounded in norm, the two methods do not give rise to significant differences in practice. A few simulation examples relative to the two approaches are reported in the next section.

D. Simulation results

The proposed strategy has been tested both via numerical simulations and experiments with a non-holonomic mobile robot. The simulations have been performed via an ad-hoc software simulator written in C language that uses the same parameters that characterize the experimental mission like, e.g., robot kinematic model parameters, control algorithm execution sample time, the NSB gains, the obstacle safety distance and the actuator saturation values. The chosen parameters are summarized in the following table:

<table>
<thead>
<tr>
<th>Kinematic parameters</th>
<th>Mission parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r = 8mm$</td>
<td>$\lambda_1 = A_2 = 1$</td>
</tr>
<tr>
<td>$b = 2.65cm$</td>
<td>$v_{max} := \max</td>
</tr>
<tr>
<td>$\frac{\Delta_p}{b} = 1$</td>
<td>$\omega_{max} := \max</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{1,d} = 40cm$</td>
</tr>
</tbody>
</table>

In the proposed mission, the robot, starting from a position close to a punctual obstacle, has to sequentially reach two target configurations. Figure 2 shows the path followed by the robot and the obstacle avoidance safety region (in dotted red) in the two saturation cases: saturation applied at velocity commands $\bar{v}_1$ (in blue) and saturations of wheels’ speeds $\omega_j$ (in green, notice the paths are almost perfectly overlapping).

Since the starting position is deeply inside the obstacle avoidance safety region, at the beginning of the mission the obstacle avoidance task function saturates the velocity request; thus, the move-to-goal task is deleted by the scaling factor and the robot starts its movement escaping (almost) radially with respect to the circle centered in the obstacle. Once sufficiently far from the obstacle, the vehicle heads toward the first goal, exiting along the tangent to the circle. After the achievement of goal 1, the robot tries to reach the second goal, behind the obstacle. Yet once again, during its mission the robot meets the obstacle and then turns around the obstacle avoidance safety region (sliding on the null-space of the obstacle avoidance avoidance region).

Figure 3 shows the norms of task velocity commands $\bar{v}_j$ relative to the first saturation case. Figure 3.a-b respectively show the plots of the norm of $\bar{v}_j$ elaborated by the NSB
as defined in (8) and after the multiplication by the scaling factors $v_1$ (in red) and $v_2$ (in green). It is worth noticing from figure 3.b that the norms of all velocity commands are always under the saturation threshold $v_{max}$, and, as the figure 3.c shows, this property is also true for their vectorial sum.

The robot can communicate through a Bluetooth module with a remote Linux-based PC where the NSB has been implemented. The low level wheel velocity controller on board of the robot is a PID developed by the manufacturer. The encoders resolution is such that a quantization of $\approx \frac{0.8}{cm/s}$ and $\approx \frac{9}{deg/s}$ are experienced. Figure 7 shows the plot of the wheels' velocities in $\text{rad/s}$ and $\text{rad/s}$, respectively. Figure 8 shows the main saturation parameters during the first goal of the experiment, $v_0$ is lower than 1 and it saturates the velocity to $v_{max}$ of the robot through the Bluetooth module. The low level wheel velocity controller sends the position measurements at a sampling time of $70\text{ms}$ and elaborates the NSB control. Once the NSB outputs the desired linear velocities, the wheels' desired velocities are elaborated referring to the kinematic model of Section IV-B and sent to the robot. The capacity $c$ of the secondary task is $\approx \frac{1}{deg/s}$, thus the scaling factor $s_1$ is lower than 1 and it saturates the velocity to $v_{max}$ (see fig. 8.b-c). In the meanwhile, the capacity $c_2$ of the secondary task is null and the scaling factor $s_2$ completely cancels the velocity of the secondary task $v_2$. When the output of the primary

E. Experimental results

The experimental platform is based on a Khepera II mobile robot manufactured by K-team [1] and available at the LAI (Laboratorio di Automazione Industriale) of the Università degli Studi di Cassino (see Figure 1). This robot is a differential-drive mobile robot (with a unicycle-like kinematics) with an approximate dimension of $8\text{cm}$ of diameter. The robot can communicate through a Bluetooth module with a remote Linux-based PC where the NSB has been implemented. To allow the needed absolute position measurements we have developed a vision-based system using two CCD cameras, a Matrox Meteor-II frame grabber [2] and self-developed C++ image-processing functions. The acquired images are $1024 \times 768$ RGB bitmaps. The measurement error has an upper bound of $\approx \frac{0.5}{cm}$ and $\approx \frac{1}{deg}$. The remote PC receives the position measurements at a sampling time of $70\text{ms}$ and elaborates the NSB control. Once the NSB outputs the desired linear velocities, the wheels' desired velocities are elaborated referring to the kinematic model of Section IV-B and sent to the robot through the Bluetooth module. The low level wheel velocity controller on board of the robot is a PID developed by the manufacturer. The encoders resolution is such that a quantization of $\approx \frac{0.8}{cm/s}$ and $\approx \frac{9}{deg/s}$ are experienced.

Analogously to the simulative case study, Figure 6 shows the paths of the mobile robot during the overall mission for two experiments with saturation at the velocity commands and at the wheels' speed. Moreover, a video of one of the experiments accompanies this paper. The behaviors of the numerical simulations and of the experiments are in good agreement; however, the noise and the light irregularity of the robot path is due to several experimental factors, e.g., a relatively high sample time of the control algorithm execution (that is performed on a remote controller), the quantization due to the encoders resolutions and to hardware/software limits of the Khepera II mobile robot, measurement noise of the vision system for position estimation. Moreover, the difference in the mission execution time is due to the vehicle dynamics that in the simulation cases is neglected.

Figure 7 shows the plot of the wheels’ velocities in both the saturation cases. In particular, figure 4.a-b respectively show the saturation applied to the velocity commands $v_j$ and wheels’ speeds $\omega_j$ while figure 3.c shows the norm of the velocity vector $v_j$; it is worth noticing that the robots mostly move at the highest allowed speed during all the mission. Finally, figure 5 shows the behavior of the robot linear velocity norm in both the saturation cases.

![Fig. 4. Wheels’ angular velocities $\omega_j$ (in blue) and $\omega_r$ (in red) during the simulation.](image1)

![Fig. 5. Norms of linear velocities during the simulation with saturation applied at $v_j$ (in blue) and at $\omega_j$ (in green).](image2)

![Fig. 6. Path followed by the robot during two experiments: one with saturations applied to the velocity commands $v_j$ (blue path) and the other with saturation to the wheels’ velocities $\omega_j$ (green path). The dotted red line represents the safe area of the obstacle.](image3)
task $\bar{v}_1$ goes under the threshold, the scaling factor $s_1$ is constantly equal to 1 while the scaling factor $s_2$ smoothly activates the secondary task (keeping $\|\bar{v}_d\|$ to the saturation value).

Fig. 7. Wheels’ angular velocities $\omega_t$ (in blue) and $\omega_r$ (in red) during the experiment. a) Case of saturations applied to the velocity commands $\bar{v}_j$; b) Case of saturations applied to the wheels’ speed $\omega_j$.

Fig. 8. Quantities involved into the saturation management technique with velocity commands $\bar{v}_j$ saturation during the first 6 seconds of the experiment. a) Norm of $\bar{v}_1$ (in red) and capacity $c_1$ (in dash dotted magenta); b) Scaling factor $s_1$; c) Norm of $\bar{v}_1 s_1$ (in red) and capacity $c_1$ (in dash dotted magenta); d) Norm of $\bar{v}_2$ (in green) and capacity $c_2$ (in dash dotted magenta); e) Scaling factor $s_2$; f) Norm of $\bar{v}_2 s_2$ (in green) and capacity $c_2$ (in dash dotted magenta) result superimposed.

V. CONCLUSIONS

In this paper, the combined application of a behavior-based technique, namely the Null-Space based Behavioral control [4], with a saturation management technique [13] has been investigated. The scope of this paper was to extend the Null-Space based Behavioral control to the motion control of robotic systems with velocity saturated actuators, avoiding that velocity saturations induced by lower priority tasks corrupt the higher priority ones. In particular, the proposed solution aims at managing actuator velocity saturations by dynamically scaling task velocity commands so that the hierarchy of task priorities is preserved in spite of actuator velocity constraints. The proposed approach has been tested on the motion control of a single non-holonomic mobile robot that has to reach a target avoiding a punctual obstacle. The approach has been validated both by numerical simulations and experimental results with a non-holonomic mobile robot.

REFERENCES


