2018 Second IEEE International Conference on Robotic Computing

# Laser-Based Control Law For Autonomous Parallel And Perpendicular Parking

David Pérez-Morales*, Olivier Kermorgant*, Salvador Domínguez-Quijada* and Philippe Martinet†
*ARMEN Team
École Centrale de Nantes - LS2N
1 rue de la Noë, 44321 Nantes, France
Emails: David.PerezMorales@ls2n.fr, Olivier.Kermorgant@ec-nantes.fr, Salvador.DominguezQuijada@ls2n.fr
†ARMEN Team
INRIA Sophia Antipolis and École Centrale de Nantes - LS2N
Emails: Philippe.Martinet@inria.fr, Philippe.Martinet@ls2n.fr

***Abstract*—This paper addresses the perpendicular and parallel parking problems of car-like vehicles for both forward and reverse maneuvers in one trial by extending the work presented in [1] using a multi sensor-based controller with a weighted control scheme. The perception problem is discussed briefly considering a Velodyne VLP-16 and a SICK LMS151 as the sensors providing the required exteroceptive information. The results obtained from simulations and real experimentation for different parking scenarios show the validity and potential of the proposed approach. Furthermore, it is shown that, despite the need of handling several constraints for collision avoidance, the required computation time of the proposed approach is small enough to be used online.**

***Keywords*-autonomous vehicles; sensor-based control; wheeled robots maneuvering;**

## I. Introduction

Even for experienced drivers, parking can be a difficult task, especially in big cities where the parking spots are often very narrow. The search for an increase in comfort and safety when parking has led to a quite extensive literature [2], having explored many different approaches to automate this bothersome task.

Despite the fact that the automobile industry has already started to roll out some commercial implementations of active parking assistants capable of actively controlling acceleration, braking and steering [3], the research interest in the topic remains strong.

Path planning approaches have been heavily investigated in recent years. Among the different planning techniques it is possible to distinguish between geometric approaches, with either constant turning radius [4], [5] using saturated feedback controllers, or continuous-curvature planning using clothoids [6]; heuristic approaches [7] and machine learning techniques [8]. It is worth noting that parking maneuvers with forward motions are seldom considered, with [9] for the parallel parking case and [10] for the perpendicular case being some of the few works on this regard.

A well-known drawback of path planning and tracking is its dependence on the localization performance. An interesting alternative that does not rely on the localization is the use of a sensor-based control approach. It has been proven to be valid for navigation [11], dynamic obstacle avoidance [12], and for parking applications [1].

The contribution of this paper is an extension of the approach described in [1], this time considering multiple sensors, a better suited sensor feature set that allows to park in one maneuver not only in perpendicular spots but also in parallel ones with either reverse or forward motions with only some minor changes, and improved constraints for collision avoidance.

In the next section the models considered as well as the notation used are presented. In Section III the perception problem is briefly addressed, showing how the sensor data are processed in order to extract the empty parking spot to latter in Section IV describe the interaction model and how to extract the required sensor features from the computed empty parking spot. Afterwards, the controller is presented in Section V and the obtained results from simulation and real experimentation for different parking scenarios are shown in Section VI. Finally, some conclusions are given in Section VII.

## II. Modeling and notation

Given that parking maneuvers are low-speed motions, a kinematic model can be considered as accurate enough.

### A. Car-like robot model and notation

The kinematic model considered is the one used to represent a car with rear-wheel driving:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \tan\phi/l_{wb} \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \dot{\phi}, \qquad (1)$$

where $v$ and $\dot{\phi}$ are the longitudinal and steering velocities.

The point M is located at the mid-distance between the passive fixed wheels (rear) axle and the distance between the rear and the front axle is described by $l_{wb}$. The generalized coordinates are $\mathbf{q} = [x, y, \theta, \phi]^T$ where $x$ and $y$ are the Cartesian coordinates of the point M, $\theta$ is the orientation of the platform with respect to the $x_0$ axis and the steering angle of the steerable wheel(s) is described by $\phi$ (Fig. 1a).

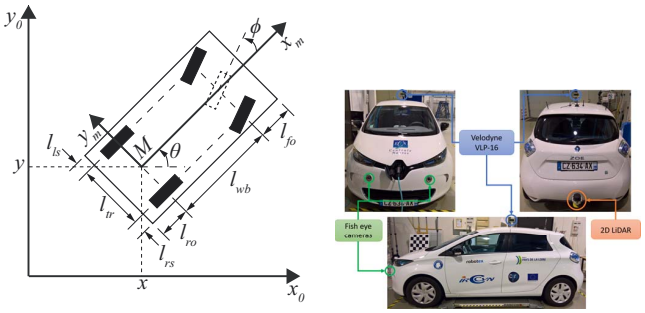
0-7695-6370-8/18/31.00 ©2018 IEEE
DOI 10.1109/IRC.2018.00018

TABLE I: Parameters

| *Parameteres* | *Notation* | *Value* |
|---|---|---|
| Wheelbase: Distance between the front and rear wheel axles | $l_{wb}$ | 2.588 m |
| Rear overhang: Distance between the rear wheel axle and the rear bumper | $l_{ro}$ | 0.657 m |
| Maximum steering angle | $\phi_{max}$ | 30° |
| Total length of the vehicle | $l_{ve}$ | 4.084 m |
| Total width of the vehicle | $w_{ve}$ | 1.945 m |
| Maximum (desired) longitudinal velocity | $v_{max}$ | 2 km/h |
| Maximum acceleration increment | $\Delta_{acc}$ | $\text{sign}(v_{n-1})\ 0.2\ T_s$ |
| Maximum deceleration increment | $\Delta_{dec}$ | $\text{sign}(v_{n-1})\ 2.5\ T_s$ |
| Maximum $\phi$ increment | $\Delta_\phi$ | $2°\ T_s$ |

Table I presents the different parameters used in the paper.

The vehicle used for experimentation and simulation is a Renault ZOE (Fig. 1b). It is represented by its bounding rectangle.




(a) Kinematic model diagram (b) Robotized Renault ZOE

Fig. 1: Kinematic model diagram for a car-like rear-wheel driving robot and vehicle used for simulation and real experimentation

### B. Multi-sensor modeling

Following our previous work [1], where a novel sensor-based control technique based on the framework described in [13] was proposed, in this paper we explore a different sensor features set to park in one maneuver into perpendicular and parallel spots considering multiple sources for the sensor signals.

*1) Kinematic model:* Let us consider a robotic system equipped with $k$ sensors (Fig. 2) that provide data about the robot pose in its environment. Each sensor $S_i$ gives a signal (sensor feature) $\mathbf{s}_i$ of dimension $d_i$ with $\sum_{i=1}^{k} d_i = d$.

In a static environment, the sensor feature derivative can be expressed as follows:

$$\dot{\mathbf{s}}_i = \check{\mathbf{L}}_i \check{\mathbf{v}}_i = \check{\mathbf{L}}_i {}^i\check{\mathbf{T}}_m \check{\mathbf{v}}_m \tag{2}$$

where $\check{\mathbf{L}}_i$ is the interaction matrix of $\mathbf{s}_i$ and ${}^i\check{\mathbf{T}}_m$ is the screw transformation matrix that allows to express the sensor twist $\check{\mathbf{v}}_i$ with respect to the robot twist $\check{\mathbf{v}}_m$.

Denoting $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_k)$ the $d$-dimensional signal of the multi-sensor system, the signal variation over time can be linked to the moving vehicle twist:

$$\dot{\mathbf{s}} = \check{\mathbf{L}}_\mathbf{s} \check{\mathbf{v}}_m \tag{3}$$

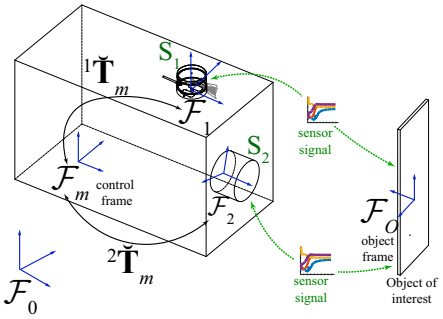



Fig. 2: Multi-sensor model

with:

$$\check{\mathbf{L}}_\mathbf{s} = \check{\mathbf{L}}\check{\mathbf{T}}_m = \begin{bmatrix} \check{\mathbf{L}}_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \check{\mathbf{L}}_k \end{bmatrix} \begin{bmatrix} {}^1\check{\mathbf{T}}_m \\ \vdots \\ {}^k\check{\mathbf{T}}_m \end{bmatrix} \tag{4}$$

*a) Planar world assumption:* Assuming that the vehicle to which the sensors are rigidly attached to evolves in a plane and that the sensors and vehicle have vertical parallel $z$ axes, all the twists are reduced to $[v_x, v_y, \dot{\theta}]^T$ hence the reduced forms $\check{\mathbf{L}}$, $\check{\mathbf{L}}_\mathbf{s}$, $\check{\mathbf{L}}_i$, $\check{\mathbf{v}}_m$ and ${}^i\check{\mathbf{T}}_m$ of, respectively, $\check{\mathbf{L}}$, $\check{\mathbf{L}}_\mathbf{s}$, $\check{\mathbf{L}}_i$, $\check{\mathbf{v}}_m$ and ${}^i\check{\mathbf{T}}_m$ are considered.

$\check{\mathbf{L}}_i$ is of dimension $d_i \times 3$, $\check{\mathbf{v}}_m = [v_{x_m}, v_{y_m}, \dot{\theta}]^T$ and ${}^i\check{\mathbf{T}}_m$ is defined as:

$${}^i\check{\mathbf{T}}_m = \begin{bmatrix} \mathrm{c}({}^m\theta_i) & \mathrm{s}({}^m\theta_i) & t_{x_i}\mathrm{s}({}^m\theta_i) - t_{y_i}\mathrm{c}({}^m\theta_i) \\ -\mathrm{s}({}^m\theta_i) & \mathrm{c}({}^m\theta_i) & t_{x_i}\mathrm{c}({}^m\theta_i) + t_{y_i}\mathrm{s}({}^m\theta_i) \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

where ${}^m\mathbf{t}_i = [t_{x_i}, t_{y_i}]^T$ and ${}^m\theta_i$ are, respectively, the position and orientation of $S_i$ with respect to $\mathcal{F}_m$ expressed in $\mathcal{F}_m$, with $\mathrm{c}({}^m\theta_i) = \cos({}^m\theta_i)$ and $\mathrm{s}({}^m\theta_i) = \sin({}^m\theta_i)$.

Furthermore, since in the considered model the control frame $\mathcal{F}_m$ is attached to the vehicle's rear axis with origin at the point $M$ (Fig. 1a), it is not possible to generate a velocity along $y_m$ on the vehicle's frame and assuming that there is no slipping nor skidding (i.e. $v_{y_m} = 0$), the robot twist $\check{\mathbf{v}}_m$ can be further reduced to:

$$\mathbf{v}_m = [v_{x_m}, \dot{\theta}]^T \tag{6}$$

with $v_{x_m} = v$, thus it is possible to write:

$$\dot{\mathbf{s}} = \mathbf{L}_\mathbf{s}\, \mathbf{v}_m \tag{7}$$

where $\mathbf{L}_\mathbf{s}$ is composed of the first and third columns of $\check{\mathbf{L}}_\mathbf{s}$.

It is worth noting that this planar world assumption is not a limitation of the presented approach but rather a tool that allows to simplify the different equations and matrices taking advantage of some of the properties of the our current configuration (parallel $z$ axes). For a more general case one could consider the following relation:

$$\dot{\mathbf{s}} = \mathbf{L}_\mathbf{s}\, \mathbf{v}_m + \check{\mathbf{L}}_\mathbf{s}^{unc} \check{\mathbf{v}}_m^{unc}$$

where $\check{\mathbf{v}}_m^{unc}$ is composed of the uncontrolled motions (extracted from $\check{\mathbf{v}}_m$) and with $\check{\mathbf{L}}_\mathbf{s}^{unc}$ being its interaction matrix.

*2) Weighted error:* The weighted multi-sensor error signal is defined as:

$$\mathbf{e_W} = \mathbf{We} \tag{8}$$

where $\mathbf{e} = \mathbf{s} - \mathbf{s}^*$ is the difference between the current sensor signal $\mathbf{s}$ and its desired value $\mathbf{s}^*$ and $\mathbf{W}$ is a diagonal positive semi-definite weighting matrix that depends on the current value of $\mathbf{s}$. Its associated interaction matrix is $\check{\mathbf{L}}_{\mathbf{W}} = \mathbf{W}\check{\mathbf{L}}_{\mathbf{s}}$.

## III. Perception

We focus the perception on the detection of parked cars. They can be approximated by boxes considering that, when viewed from the top, have a rectangular-like shape.

The vehicle used (Fig. 1b) has been equipped with many sensors (Velodyne VLP-16, SICK LMS151, GPS, cameras in the front, etc.) to observe its environment, a computer to process the data and actuators that can be computer controlled. Since our application requires exteroceptive information from all around the vehicle at, potentially close distances, the VLP-16 and SICK LMS151 were the sensors chosen to work with.

Because both sensors provide information of a very similar nature, the data can be fused by simply storing the data provided by the LMS151 as a point cloud and then transforming the point cloud from LMS151's frame to the VLP-16's frame so it can be added to the point cloud provided by the latter sensor. For this, it is assumed that the time difference between the data provided by each sensor is reasonably small, i.e. the data are sufficiently synchronized.

The complete point cloud obtained from the two sensors is first filtered with a couple of crop boxes. The first crop box keeps only the data that are close enough to the car to be relevant in a parking application and that does not represent the floor and afterwards and the second one is used to filter out the points that belong to the car's body (self-collision sensor readings). Then, an Euclidean Cluster Extraction algorithm is used to have each obstacle represented as a cluster.

The orientation of each cluster is extracted by fitting a line model to the points belonging to the contour of the cluster using a RANSAC algorithm. The orientation of the bounding box will be equal to the orientation of the fitted line. After, we proceed by finding the rotated bounding box of the cluster using the previously found orientation. We have found from experience that this method gives better results than the one included (getOBB) in the Point Cloud Library (PCL) which is based on the eigenvectors of the cluster.

The empty parking place (green rectangle in Fig. 3) is extracted using the approach described in [1]. The sensor features required for the controller are extracted from this computed parking place.

Considering that the empty parking place is defined in the Cartesian space, it is possible to use virtual sensors placed at will that give the required sensor features in a convenient frame. Indeed, thanks to the use of virtual sensors, different physical sensors may be used as long as the necessary sensor features can be computed.

## IV. Interaction model

The interaction model relies on the perception of several lines ${}^i\mathcal{L}_j$ (parametrized using normalized Plücker coordinates) and points ${}^ip_a({}^ix_a, {}^iy_a)$ from several (virtual) sensors $S_i$.

The sensor's placement can be seen in Fig. 3. The placement of $S_1$ and $S_2$ correspond, respectively, to the VLP-16's and LMS151's. $S_3$ to $S_6$ are placed on the corners of the car's bounding rectangle (taking into account the side mirrors) and have the same orientation as $\mathcal{F}_m$

To illustrate the feature extraction approach, the case of a reverse perpendicular maneuver is now detailed. As it can be seen in Fig. 3, points ${}^ip_1$ to ${}^ip_4$ correspond to the corners of the parking spot while ${}^ip_5$ and ${}^ip_6$ are, respectively, the midpoints between $({}^ip_1, {}^ip_4)$ and $({}^ip_2, {}^ip_3)$. ${}^i\mathcal{L}_1$ is a line that passes through ${}^ip_5$ and ${}^ip_6$, i.e. it passes through the center of the parking spot. ${}^i\mathcal{L}_2$ is a line that passes through ${}^ip_1$ and ${}^ip_4$ thus corresponding to the depth limit of the parking spot. ${}^i\mathcal{L}_3$ is a line that passes through ${}^ip_3$ and ${}^ip_4$ which would represent the left boundary of the parking spot.

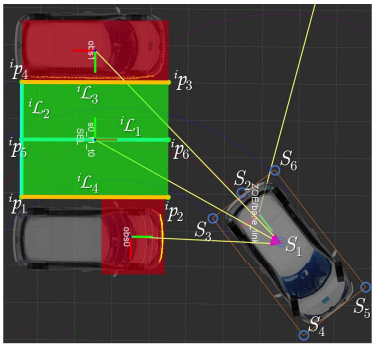



Fig. 3: Sensors' configuration and sensor features

The exact definition of the sensor features varies depending on the parking scenario although in any case, ${}^i\mathcal{L}_1^*$ should be collinear with $x_m$ and ${}^i\mathcal{L}_2^*$ should be parallel to $y_m$ and be behind the car for reverse maneuvers and in front for forward ones. In this paper we only detail the actual features used for a specific case, but deducing the features that should be used for other cases isn't complicated.

Considering the previously mentioned assumption that the vehicle to which the sensors are attached to evolves in a plane (sensors and vehicle with parallel $z$ axes), the sensor signal $\mathbf{s}_{i_{\mathcal{L}_j}}$ and interaction matrix $\check{\mathbf{L}}_{i_{\mathcal{L}_j}}$ for the line ${}^i\mathcal{L}_j$ observed by $S_i$ are defined respectively as:

$$\mathbf{s}_{i_{\mathcal{L}_j}} = \left[{}^i\underline{\mathbf{u}}_j(1), {}^i\underline{\mathbf{u}}_j(2), {}^i\mathbf{h}_j(3)\right]^T \tag{9}$$

$$\check{\mathbf{L}}_{i_{\mathcal{L}_j}} = \begin{bmatrix} 0 & 0 & {}^i\underline{\mathbf{u}}_j(2) \\ 0 & 0 & -{}^i\underline{\mathbf{u}}_j(1) \\ -{}^i\underline{\mathbf{u}}_j(2) & {}^i\underline{\mathbf{u}}_j(1) & 0 \end{bmatrix} \tag{10}$$

where ${}^i\underline{\mathbf{u}}_j = {}^i\mathbf{u}_j/||{}^i\mathbf{u}_j||$ with ${}^i\mathbf{u}_j \neq 0$ denoting the orientation of $\mathcal{L}_j$ and ${}^i\mathbf{h}_j = {}^i\mathbf{r}_j/||{}^i\mathbf{u}_j||$ with ${}^i\mathbf{r}_j$ containing the coefficients of the interpretation plane equation [14]. In the 2D configuration considered, the components of ${}^i\underline{\mathbf{u}}_j$ and ${}^i\mathbf{h}_j$

that don't appear in (9) are equal to 0 thus ${}^{i}\mathbf{h}_j(3)$ can be interpreted as the distance to the line.

It should be noted that the weighting and constraints required to park safely change depending on the type of parking spot (parallel, perpendicular or diagonal) and on which side the parking spot is placed with respect to the car at the beginning of the maneuver.

### A. Task sensor features

The set of task ($^t$) sensor features $\mathbf{s}^t$ corresponding to the positioning is defined as:

$$\mathbf{s}^t = [\mathbf{s}_{i_{\mathcal{L}_1}}, \mathbf{s}_{i_{\mathcal{L}_2}}]^T \tag{11}$$

with $\mathbf{s}^t$ obtained from $S_1$ for forward maneuvers and from $S_2$ for reverse ones.

The corresponding interaction matrix $\check{\mathbf{L}}^t$ is computed by a 2nd order approximation of the form:

$$\check{\mathbf{L}}^t = \frac{\check{\mathbf{L}}_{\mathcal{L}} + \check{\mathbf{L}}^*_{\mathcal{L}}}{2} \tag{12}$$

where $\check{\mathbf{L}}_{\mathcal{L}} = [\check{\mathbf{L}}_{i_{\mathcal{L}_1}}, \check{\mathbf{L}}_{i_{\mathcal{L}_2}}]^T$ and $\check{\mathbf{L}}^*_{\mathcal{L}}$ is equal to the value of $\check{\mathbf{L}}_{\mathcal{L}}$ at the desired pose.

The associated weighting matrix $\mathbf{W}^t$ is defined by (13). The values of $w_3^t$ and $w_6^t$ are constant while the values of $w_i^t \ \forall i = 1, 2, 4, 5$ are computed using a smooth weighting function (Fig. 4) based on the one presented in [15]:

$$\mathbf{W}^t = \text{diag}(w_1^t, w_2^t, w_3^t, w_4^t, w_5^t, w_6^t) \tag{13}$$

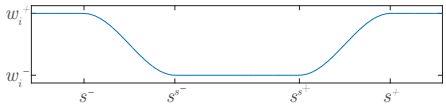



Fig. 4: Weighting function $w_i^t$

With this weighting function and considering the weighting parameters (31), $s_i \ \forall \ i = \{2, 4, 5\}$ could be seen as bounded constraints but in fact they are task features, given that we do care about the value of their corresponding $\mathbf{e}^t(i)$. If only $s_3$ and $s_6$ were considered as task features, the car may finish the maneuver with a bad orientation even if $\mathbf{e}^t(3) \approx \mathbf{e}^t(6) \approx 0$ because just 2 features are not enough to control the car's DOFs.

Due to space constraints, only the case of a reverse perpendicular parking maneuver with the spot placed on the right will be considered for the rest of this section.

### B. Constrained sensor features

For a reverse perpendicular parking maneuver, the set of constrained sensor features $\mathbf{s}^c$ used for collision avoidance is defined as:

$$\mathbf{s}^c = [\mathbf{s}_3, \mathbf{s}_5, \mathbf{s}_6]^T \tag{14}$$

To define the constraints, two other types of features are considered in addition to ${}^{i}\mathcal{L}_j$: the $y$ coordinate of a given point (${}^{i}y_a$) and the difference of radii (shown in Fig. 5):

$${}^{i}d_{lat_a} = {}^{i}\rho_{p_a} - \rho_{lat} \tag{15}$$

where:

$${}^{i}\rho_{p_a} = \sqrt{({}^{i}x_a + t_{x_i})^2 + ({}^{i}y_a + t_{y_i} - \rho_m)^2} \tag{16}$$

$$\rho_{lat} = |\rho_m| - \frac{w_{ve}}{2} \tag{17}$$

with $\rho_m = l_{wb}/\tan\phi$.

Since, as stated before, the constraints are used for collision avoidance, we are interested only in the components of (9) related to the distance to the feature itself, therefore:

$$\mathbf{s}_3 = [{}^{3}\mathbf{h}_2(3), {}^{3}y_2, {}^{3}d_{lat_2}]^T \tag{18}$$

$$\mathbf{s}_5 = {}^{5}\mathbf{h}_3(3) \tag{19}$$

$$\mathbf{s}_6 = [{}^{6}\mathbf{h}_2(3), {}^{6}\mathbf{h}_3(3)]^T \tag{20}$$

The corresponding interaction matrices are:

$$\check{\mathbf{L}}_3 = \begin{bmatrix} -{}^{3}\underline{\mathbf{u}}_2(2) & {}^{3}\underline{\mathbf{u}}_2(1) & 0 \\ 0 & -1 & -{}^{3}x_2 \\ 0 & \frac{{}^{3}\varrho_y}{{}^{3}\rho_{p_2}^2} & \frac{{}^{3}x_2\,{}^{3}\varrho_y}{{}^{3}\rho_{p_2}^2} \end{bmatrix} \tag{21}$$

$$\check{\mathbf{L}}_5 = \begin{bmatrix} -{}^{5}\underline{\mathbf{u}}_3(2) & {}^{5}\underline{\mathbf{u}}_3(1) & 0 \end{bmatrix} \tag{22}$$

$$\check{\mathbf{L}}_6 = \begin{bmatrix} -{}^{6}\underline{\mathbf{u}}_2(2) & {}^{6}\underline{\mathbf{u}}_2(1) & 0 \\ -{}^{6}\underline{\mathbf{u}}_3(2) & {}^{6}\underline{\mathbf{u}}_3(1) & 0 \end{bmatrix} \tag{23}$$

with ${}^{i}\varrho_y = -|{}^{i}y_a + t_{y_i} - \rho_m|$.

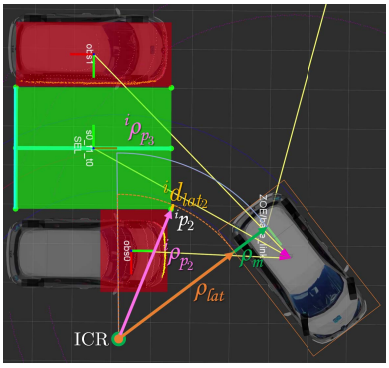



Fig. 5: Radial constraint: all the radii define concentric arcs with center at ICR (Instantaneous Center of Rotation)

It should be noted that some constraints must be deactivated under certain conditions in order to be able to park successfully. For the considered case: ${}^{3}y_2$ must be deactivated if $\phi < 0$ while ${}^{3}d_{lat_2}$ must be deactivated if $\phi \geq 0$. The reasoning behind the deactivation is that, on the one hand, if $\phi > 0$ the ICR would be located on the left side of the vehicle rendering the constraint on ${}^{3}d_{lat_2}$ useless and, on the other hand, if $\phi < 0$ the constraint on ${}^{3}d_{lat_2}$ will also ensure the one on ${}^{3}y_2$ rendering the latter redundant and unnecessary.

## V. Control

When considering the constraints presented in Sec. IV-B (particularly (15)), a chattering problem appears if the controller presented in [1] is used, even with very small weights. For this reason, that controller is adapted to the quadratic programming form [16] with only inequality constraints:

$$\begin{aligned} \mathbf{v}_m = \text{argmin}||\mathbf{L}^t_{\mathbf{W}}\mathbf{v}_m + \lambda \mathbf{e}^t_{\mathbf{W}}||^2 \\ \text{s.t. } \mathbf{A}\mathbf{v}_m \leq \mathbf{b} \end{aligned} \tag{24}$$

with:

$$\mathbf{A} = [\mathbf{L}^c, -\mathbf{L}^c]^T \tag{25}$$

$$\mathbf{b} = [\alpha(\mathbf{s}^{c^+} - \mathbf{s}^c), -\alpha(\mathbf{s}^{c^-} - \mathbf{s}^c)]^T \tag{26}$$

where $\alpha$ is a gain constant, $\lambda$ is the control gain and $[\mathbf{s}^{c^-}, \mathbf{s}^{c^+}]$ is the interval in which $\mathbf{s}^c$ should remain. Since the constraints are used for collision avoidance, only one side of the interval $[\mathbf{s}^{c^-}, \mathbf{s}^{c^+}]$ has to be defined for each feature.

To solve (24), a generic solver is used. To improve the convergence and computation time, the optimization variables are $[v, \phi]$ and not $\mathbf{v}_m$, although inside the objective function $\dot{\theta}$ is computed from $\phi$ so (24) can be solved. When using $\phi$ instead of $\dot{\theta}$ one can easily take into account the steering wheel's joint limits by bounding the output with (27) at the solving step instead of solving (24) directly with $\mathbf{v}_m$ as optimization variables and hope for the value of $\phi$ computed from $\dot{\theta}$ to fall inside the bound (27).

$$|\phi| < \phi_{max} \tag{27}$$

The longitudinal velocity is saturated by:

$$|v| < v_{max} \tag{28}$$

where $v_{max}$ is an adaptive bound imposing a deceleration profile based on the velocity profile shown in [5] as the vehicle approaches the final pose.

Finally, to avoid large changes in the control signals at time instant $n$ that may cause uncomfortable sensations for the passengers or surrounding witnesses and to consider to some extent the dynamic limitations of the vehicle, the control signals are saturated by some increments with respect to the control signal at $n-1$ as shown below:

$$(v_{n-1} - \Delta_{dec}) \leq v_n \leq (v_{n-1} + \Delta_{acc}) \tag{29}$$

$$(\phi_{n-1} - \Delta_{\phi}) \leq \phi_n \leq (\phi_{n-1} + \Delta_{\phi}) \tag{30}$$

## VI. Results

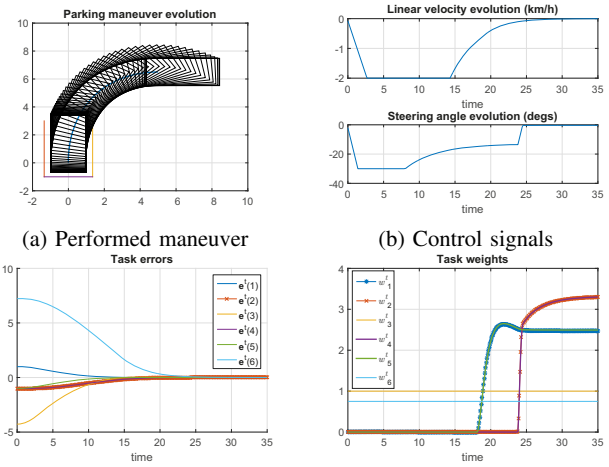
To show the potential of our approach, several parking scenarios are presented below, all of them using the final form of the controller (24). All the unconstrained cases and the first contrained case were computed in MATLAB, using fmincon as solver. For the constrained cases on the fast prototyping environment and real experimentation, NLopt with the SLSQP algorithm was used.

As an example of the weighting approach, for the case of a reverse perpendicular parking maneuver with the parking spot placed on the right, $w_i^+ = 5$, $w_i^- = 0$, $w_3^t = 1$, $w_6^t = 0.75$ and:

$$\begin{cases} s_1^{s^+} = s_1^+ = \infty \\ s_1^{s^-} = 0.001 + s_1^* \\ s_1^- = -0.001 + s_1^* \\ s_i^{s^-} = s_i^- = -\infty & \forall\ i = \{2,4,5\} \\ s_i^{s^+} = -0.001 + s_i^* & \forall\ i = \{2,4,5\} \\ s_i^+ = 0.001 + s_i^* & \forall\ i = \{2,4,5\} \end{cases} \tag{31}$$

These weighting parameters allow to prioritize the error in position over the orientation for the most part of the maneuver and, when ${}^i\underline{\mathbf{u}}_j^*(a)$ is almost reached, smoothly increase the corresponding weights so we can gradually switch the priority from positioning the vehicle to orientate it to avoid finishing the maneuver with a bad orientation.

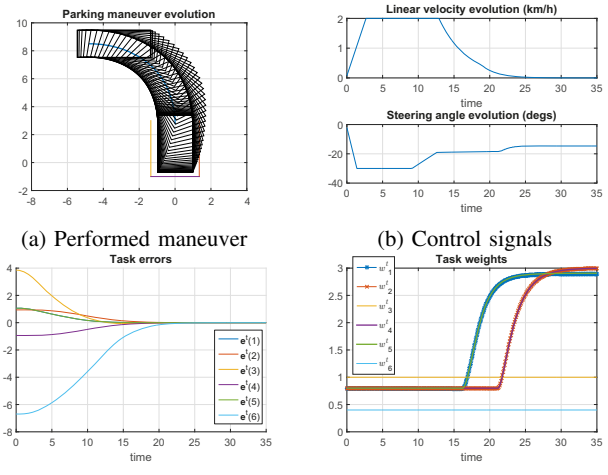
### A. Unconstrained cases - MATLAB




(a) Performed maneuver (b) Control signals

(c) Task error signal (d) Task features' weights

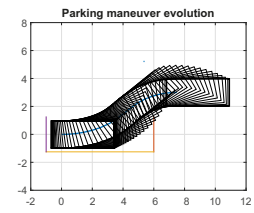
Fig. 6: Case 1: Unconstrained ⊥ reverse parking maneuver



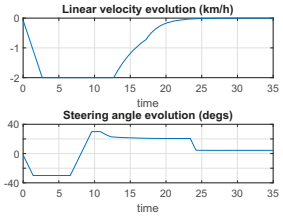

(a) Performed maneuver (b) Control signals

(c) Task error signal (d) Task features' weights

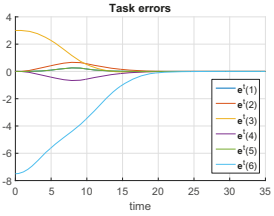
Fig. 7: Case 2: Unconstrained ⊥ forward parking maneuver



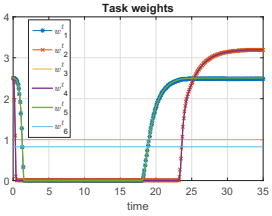

(a) Performed maneuver




(b) Control signals



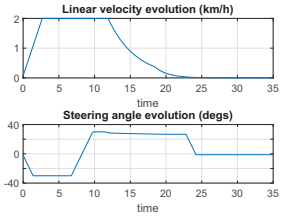

(c) Task error signal



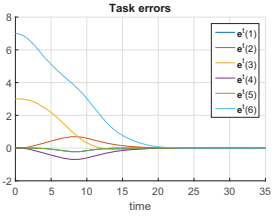

(d) Task features' weights

Fig. 8: Case 3: Unconstrained ∥ reverse parking maneuver

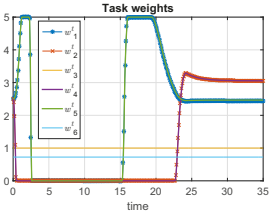

(a) Performed maneuver




(b) Control signals

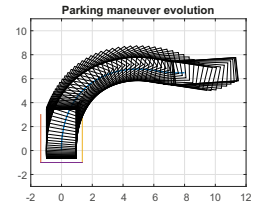



(c) Task error signal

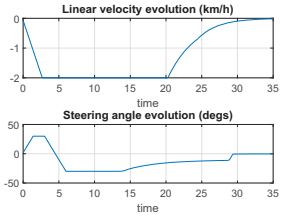



(d) Task features' weights

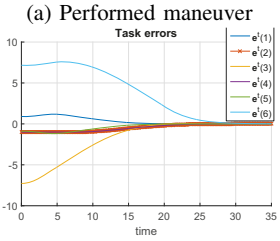
Fig. 9: Case 4: Unconstrained ∥ forward parking maneuver



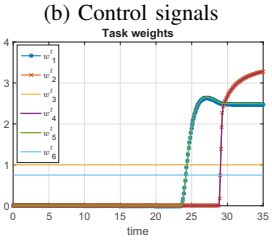

(a) Performed maneuver




(b) Control signals




(c) Task error signal




(d) Task features' weights

Fig. 10: Case 5: Unconstrained ⊥ reverse parking maneuver from far

To evaluate the performance of the proposed approach, it was first tested in unconstrained cases with a sampling time $T_s = 0.1$. As it can be seen in Figs. 6-10, the presented technique allows to perform parking maneuvers for many different scenarios (perpendicular and parallel with either reverse or forward motions) just by adjusting the weighting parameters and the specific definition of the sensor features. The final errors for all of these cases are in the order of $\times 10^{-3}$ or smaller.

Unlike our previous work [1], which required to perform gain-tunning for different initial conditions, this newly presented approach allows to park successfully for different (reasonable) initial positions and orientations of the same parking case using the same weighting parameters, although it should be mentioned that the stability, specially when constraints are considered, is still under study.

It can be seen how, for all shown cases, the weights (Figs. 6d-10d) push $\phi$ towards 0 (Figs. 6b-10b) once ${}^i\underline{\mathbf{u}}_j^*(a)$ is almost reached when close to the completion of the maneuver to keep the orientation close to the desired value.

In Fig. 10, it can be seen how even if the car is placed considerably farther from the parking spot than in Fig. 6 and not exactly perpendicular to the spot, the vehicle is able to park correctly using the same weighting parameters, showing the stability of the presented approach.

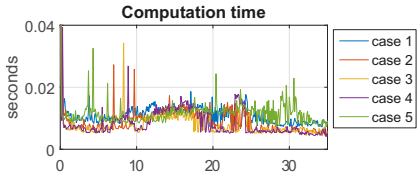



Fig. 11: Computation time

Furthermore, it can be seen in Fig. 11 that the computation time for all the unconstrained cases, with the exception of the first iteration which is around 0.35s, is mostly lower than 0.02s and remains below 0.04s all the time. Even if the computation time of the first iteration is higher than the sampling time, it does not represent a problem for our approach to be used online since at the first iteration the vehicle is not in motion. Moreover, it is safe to assume that the use of a compiled language (such as C++) would lead to faster computations than the interpreted MATLAB language.

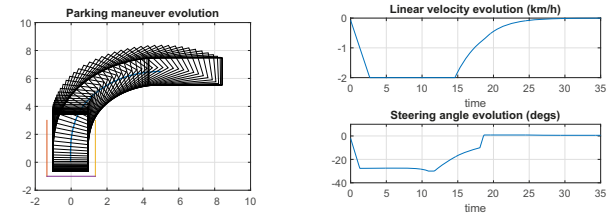
### *B. Constrained cases*

The proposed approach (considering the constraints) has been tested in three different environments: MATLAB, a homemade fast prototyping environment and in the real world.

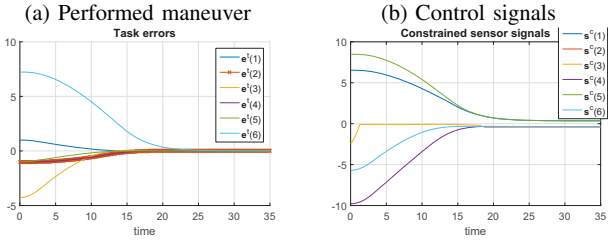
To show the performance of our approach, below we present the results obtained on each of the previously mentioned environments for a reverse perpendicular parking maneuver with the spot placed on the right. The weighting parameters remain the same as for the unconstrained case while the constraints are defined with $s_1^{c-} = 0.15$, $s_2^{c+} = -0.075$, $s_3^{c+} = -0.075$, $s_4^{c+} = -0.1$, $s_5^{c-} = 0.15$, $s_6^{c+} = -0.1$.

On the MATLAB implementation $T_s$ remains the same as for the unconstrained cases while for fast prototyping environment and real experimentation cases $T_s = 0.05$.

*1) MATLAB:* It can be seen in Fig. 12 that the vehicle is able to park successfully while satisfying the constraints during the whole maneuver, having at the end $||\mathbf{e}^t|| = 0.0317$.




(a) Performed maneuver (b) Control signals




(c) Task error signal (d) Constrained sensor signals

Fig. 12: Case 6: Constrained $\perp$ reverse parking maneuver

It should be noted that the initial position and orientation of the vehicle (Fig. 12a) is the same as for the first unconstrained case (Fig. 6a) shown in Sec. VI-A.

One thing to highlight is that, unlike for the unconstrained case (Fig. 6b), $\phi$ doesn't saturate as fast (Fig. 12b) in order to satisfy the constraint on $\mathbf{s}^c(3)$.

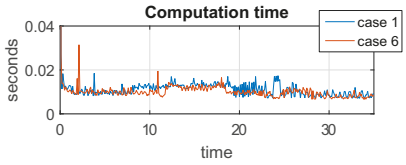



Fig. 13: Computation time

Regarding the computation time (Fig. 13), it can be seen that even when the constraints are considered it remains below 0.02s most of the time like on the unconstrained case 1.

*2) Fast prototyping environment:* A homemade fast prototyping environment using the same software architecture as the one embedded inside the car is used for simulation purposes. This homemade environment is interfaced with Gazebo to simulate the exteroceptive sensors.

As it can be seen in Figs. 14-15, the car is able to park successfully while satisfying the constraints in spite of the sensor noise and the less than perfect system response. The evolution of the many different signals, especially for the longitudinal velocity (Fig. 15a), is very similar to the unconstrained case (Fig. 6b). The fast deceleration at the end (Fig. 15a) is due to a stopping condition in the implementation related to $\mathbf{e}^t$. Regarding the evolution of $\phi$, it can be seen how, contrary to the unconstrained case, it doesn't saturate; this behavior is caused by the constraints, particularly $\mathbf{s}^c(3)$ (Fig. 15d). The final error is $\mathbf{e}^t = [-0.0003, -0.0.916, 0.0207, -0.0.916, 0.0003, 0.0569]^T$, which translates to an error in position of (2.07cm, 5.69cm) and -1.96° in orientation.

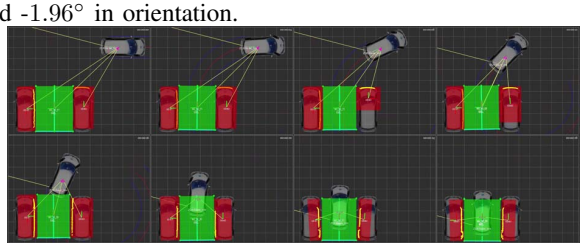
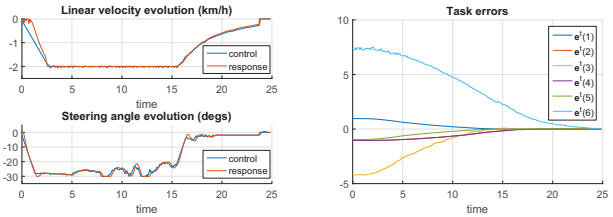
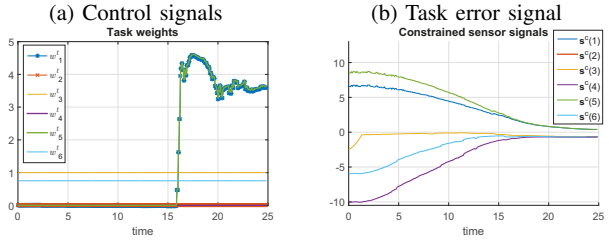


Fig. 14: Constrained perpendicular reverse parking maneuver




(a) Control signals (b) Task error signal




(c) Task features' weights (d) Constraints sensor signals

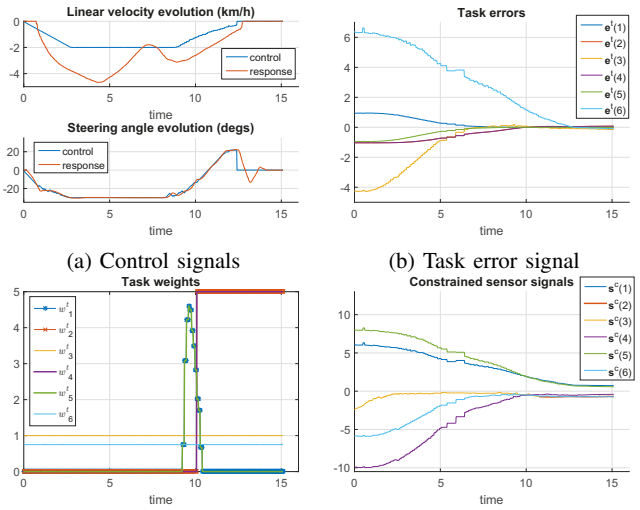
Fig. 15: Case 7: Constrained $\perp$ reverse parking maneuver signals

*3) Real experimentation:* Real experimentation was conducted for the same parking case (Fig. 16) as with the fast prototyping environment shown above. The weighting parameters and constraints definition remain the same.

Fig. 16: Experimental car parking in a perpendicular spot

It is obvious that the response of the system, particularly for the linear velocity (Fig. 17a), is less than ideal, reaching a speed more than twice as fast than what the controller indicates. This behavior can be attributed to the low-level velocity controller, which still requires some tunning to improve the performance at low velocities, therefore it has no relation to the presented technique.

Despite of the erratic response of the system in addition to the noise coming from the sensors, the constraints were respected during the whole maneuver (Fig. 17d) i.e. the vehicle remained at a safe distance from the obstacles all the time. As an example, the car got no closer than 33.88cm ($\mathbf{s}^c(6)$) to $\mathcal{L}_3$ (left boundary of the parking spot).




(a) Control signals (b) Task error signal

(c) Task features' weights (d) Constraints sensor signals

Fig. 17: Case 8: Constrained $\perp$ reverse parking maneuver signals

Furthermore, the evolution of $\mathbf{e}^t$ (Fig. 17b) is very similar to the simulated case (Fig. 15b), although the final error is not as good, being in this case $\mathbf{e}^t = [0.0054, 0.0743, -0.1436, 0.0743, -0.0054, -0.0833]^T$.

The smallest $||\mathbf{e}^t||$ was achieved at $T = 12.6399$s, with $\mathbf{e}^t = [0.0025, 0.0429, -0.0851, 0.0429, -0.0025, 0.0207]^T$, which translates to an error in position of (-8.51cm, 2.07cm) and $2.86°$ in orientation, and with $[v, \phi]^T = [0, 0]^T$ from the controller starting at $T = 12.42$s.

## VII. Conclusion

Following our previous work [1], we showed how a better choice of the sensor features allows to improve the performance, stability and versatility of the presented sensor-based approach, this time not only being able to deal successfully with perpendicular parking maneuvers but also with parallel ones with both reverse and forward motions with just some minor adjustments for each type of parking. The stability, specially when constraints are considered, is still under study.

Preliminary results obtained from real experimentation validate the robustness and effectiveness of the presented approach, considering that, despite of the erratic response of the system due to the low-level velocity controller, the car parked successfully while respecting the constraints during the whole maneuver.

Furthermore, it has been shown that even when using the interpreted MATLAB language and considering several constraints, the computation time required by our approach is small enough to be used online.

It is important to mention that, due to visibility constraints and in order to keep the results obtained with the fast prototyping environment as close to the reality as possible, only reverse parking maneuvers have been tested outside MATLAB with the presented sensor feature set. Nevertheless, the multi-sensor framework gives a high expandability, allowing for future upgrades to the perception capabilities of the system.

## Acknowledgment

This work was supported by the Mexican National Council for Science and Technology (CONACYT). This paper describes work carried out in the framework of the Valet project, reference ANR-15-CE22-0013-02.

## References

[1] D. Pérez Morales, S. Domínguez Quijada, O. Kermorgant, and P. Martinet, "Autonomous parking using a sensor based approach," in *8th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, PPNIV'16 at 19th IEEE ITSC 2016*, Rio de Janeiro, Brazil, 2016, pp. 211–216.

[2] W. Wang, Y. Song, J. Zhang, and H. Deng, "Automatic parking of vehicles: A review of literatures," *International Journal of Automotive Technology*, vol. 15, no. 6, pp. 967–978, 2014.

[3] Y. Song and C. Liao, "Analysis and Review of State-of-the-Art Automatic Parking Assist System," in *2016 IEEE International Conference on Vehicular Electronics and Safety*, Beijing, China, 2016, pp. 61–66.

[4] P. Petrov, F. Nashashibi, and M. Marouf, "Path Planning and Steering control for an Automatic Perpendicular Parking Assist System," in *7th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, PPNIV'15*, Hamburg, Germany, 2015, pp. 143–148.

[5] P. Petrov and F. Nashashibi, "Saturated Feedback Control for an Automated Parallel Parking Assist System," in *13th International Conference on Control, Automation, Robotics and Vision (ICARCV'14)*, Marina Bay Sands, Singapore, 2014, pp. 577–582.

[6] H. Vorobieva, N. Minoiu-Enache, S. Glaser, and S. Mammar, "Geometric Continuous-Curvature Path Planning for Automatic Parallel Parking," in *2013 10th IEEE International Conference on Networking, Sensing and Control (ICNSC)*, Evry, France, 2013, pp. 418–423.

[7] C. Chen, M. Rickert, and A. Knoll, "Path planning with orientation-aware space exploration guided heuristic search for autonomous parking and maneuvering," in *2015 IEEE Intelligent Vehicles Symposium*, Seoul, Korea, 2015, pp. 1148–1153.

[8] G. Notomista and M. Botsch, "Maneuver segmentation for autonomous parking based on ensemble learning," in *2015 International Joint Conference on Neural Networks (IJCNN)*, Killarney, Ireland, 2015, pp. 1–8.

[9] H. Vorobieva, S. Glaser, N. Minoiu-Enache, and S. Mammar, "Automatic parallel parking in tiny spots: Path planning and control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 396–410, 2015.

[10] K. Min and J. Choi, "A control system for autonomous vehicle valet parking," in *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*, Gwangju, South Korea, oct 2013, pp. 1714–1717.

[11] D. A. de Lima and A. C. Victorino, "Sensor-Based Control with Digital Maps Association for Global Navigation: A Real Application for Autonomous Vehicles," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Las Palmas, Spain, 2015, pp. 1791–1796.

[12] Y. Kang, D. A. de Lima, and A. C. Victorino, "Dynamic obstacles avoidance based on image-based dynamic window approach for human-vehicle interaction," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, Seoul, South Korea, jun 2015, pp. 77–82.

[13] O. Kermorgant and F. Chaumette, "Dealing with constraints in sensor-based robot control," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 244–257, 2014.

[14] N. Andreff, B. Espiau, and R. Horaud, "Visual Servoing from Lines," *International Journal of Robotics Research*, vol. 21, no. 8, pp. 679–699, 2002.

[15] V. K. Narayanan, F. Pasteau, M. Marchal, A. Krupa, and M. Babel, "Vision-based adaptive assistance and haptic guidance for safe wheelchair corridor following," *Computer Vision and Image Understanding*, vol. 149, pp. 171–185, 2016.

[16] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: generalizing the task priority framework to inequality tasks," *IEEE Trans. on Robotics*, vol. 27, no. 4, pp. 785–792, 2011.