Tentacle-based Moving Obstacle Avoidance for Omnidirectional Robots with Visibility Constraints

Abdellah Khelloufi, Nouara Achour, Robin Passama, Andrea Cherubini

Tentacle-based Moving Obstacle Avoidance for Omnidirectional Robots with Visibility Constraints

Abdellah Khelloufi[1,2], Nouara Achour[2], Robin Passama[3], Andrea Cherubini[3]

***Abstract*— This paper presents a tentacle-based obstacle avoidance scheme for omnidirectional mobile robots that must satisfy visibility constraints during navigation. The navigation task consists of driving the robot towards a visual target in the presence of environment (static or moving) obstacles. The target is acquired by an on-board camera, while the obstacles surrounding the robot are sensed by laser range scanners. To perform such task, the robot must avoid the obstacles while maintaining the target in its field of view. The approach is validated in both simulated and real experiments.**

I. Introduction

Navigation strategies generally aim at endowing a mobile robot with capacities of perception, decision, and action, that allow it to autonomously navigate in its environment. These strategies are traditionally divided in two main classes, depending on whether the problem is solved locally or globally. Indeed, the global approach [1] [2], usually consists of motion planning that relies on the knowledge of an accurate robot pose and on a global map of the environment. On the other hand, local or reactive strategies are based on instantaneous information acquired by the robot sensors. These strategies include: potential fields [3], vector field histogram [4], elastic band [5], dynamic window [6], obstacle-restriction method [7], and closest gap [8].

One of the advantages of these techniques is that they can be well combined with other sensor-based approaches such as visual servoing [9], [10]. In these cases, the task is defined in the sensor space, instead of configurations space, and it does not require neither a global model of the environment nor robot localization. Works in this area include the one presented in [11]. The authors merge an image-based controller with obstacle avoidance, for a differential-drive robot equipped with a pan actuated camera. Moving obstacles may also be considered in this approach, as shown in [12]. Recently, another interesting framework for visual navigation with obstacle avoidance has been presented in [13], for a car-like robot equipped with an actuated camera and a range scanner. The framework is based on tentacles [14], [15], i.e., drivable paths used to predict possible collisions in the near future. The task realized in [13] consists in following a visual path represented by key images, without colliding with the ground obstacles. The authors showed that obstacle avoidance had no effect on visual navigation. Furthermore, the approach presents many advantages compared to the potential fields [16]. The framework was improved in [17], by using a Kalman filter for estimating the obstacle velocities, in order to deal with moving obstacles during navigation.

In this paper, we address the problem of avoiding collisions during visual navigation of an omnidirectional mobile robot that is equipped with a fixed camera and distance sensors. Specifically, we propose a generalization of the framework from [13], by introducing *omnidirectional tentacles*. These tentacles are characterized not only by curvature, but also by course angle, since the robot linear velocity is not necessarily aligned with the robot heading, as in traditional non-holonomic systems. We also consider the visibility constraints induced by the limitations of the - fixed - camera field of view. To our knowledge, this is the first time that safe navigation with moving obstacle avoidance is carried out on omnidirectional robots with visibility constraints.

The remainder of the paper is organized as follows. In Section II, the problem and relevant variables are defined. The control scheme is presented in Sect. III. In section IV, we develop our strategy for determining the best tentacle. Simulated and real experimental results are presented in Sect.V, and we conclude in Sect. VI.

II. Problem Definition

A. General Definitions

We consider an omnidirectional robot, which can move in any direction on the ground, with control inputs:

$$\mathbf{u} = (v_X,\, v_Y,\, \omega)\,.$$

These are aligned with the axes of the robot frame $\mathscr{F}_{\mathscr{R}}(R, X_R, Y_R)$ (see Fig. 1), with R the robot center of rotation, X_R pointing forward and Y_R pointing leftward. The robot is equipped with a fixed forward looking camera with limited field of view β (so that its heading also determines its viewing direction) and distance sensors for building a local map of the obstacles surrounding it.

The task to be performed by the robot consists in driving toward a target that is seen by the camera, while avoiding the environment obstacles. We assume that the target can be visually detected and tracked by the robot. When the environment is safe, the robot should progress forward with its camera pointing at the visual target. In the case where avoidable obstacles (either static or dynamic) are present, the robot should circumnavigate

[1]Center for Development of Advanced Technologies CDTA, 20 Aout 1956 City, Baba Hassen Algiers, Algeria. akhelloufi@cdta.dz

[2]Faculty of Electronics and Computer Science, University of Sciences and Technology Houari Boumediene USTHB, BP32 EL-ALIA, 16111 Bab Ezzouar Algiers, Algeria.

[3]Laboratory for Compute Science, Micro-electronics and Robotics LIRMM - Université de Montpellier CNRS, 161 Rue Ada, 34090 Montpellier, France. lastname@lirmm.fr

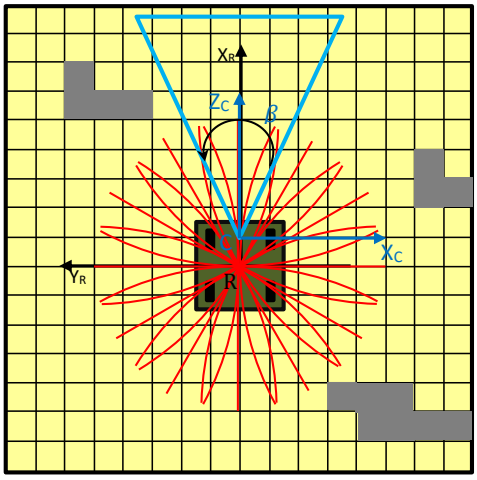

Fig. 1. General definitions.The robot frame $\mathscr{F}_{\mathscr{R}}$ is shown along with the occupancy grid (yellow) occupied cells (gray), omni-directional tentacles (red), and the camera field of view (cyan).

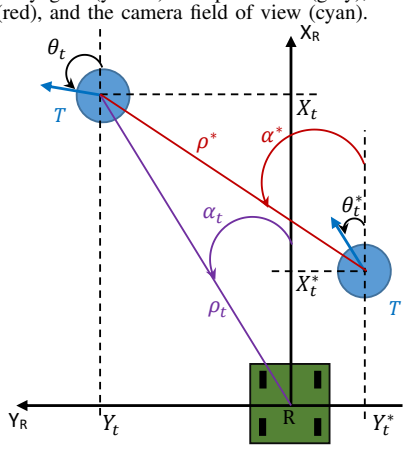

Fig. 2. Definition of the visual task variables.

them and maintain the target visibility in the camera field of view. If collision is inevitable or the target is lost, the robot should stop. The navigation task specifications are:

1) orient the robot in order to point the camera toward the target (we assume that the target is visible when the robot starts the navigation task),
2) make the robot progress toward the target,
3) avoid collision with the obstacles, while keeping the target in sight (in the camera field of view).

More formally, this consists in driving the robot so that the visual target T (that we assume static) moves from current pose ${}^r\mathbf{p}_t = [{}^rX_t\ {}^rY_t\ {}^r\theta_t]^\top$ in the robot frame $\mathscr{F}_{\mathscr{R}}$ to a desired pose, ${}^r\mathbf{p}_t^* = [{}^rX_t^*\ {}^rY_t^*\ {}^r\theta_t^*]^\top$ (see Fig. 2 for a complete illustration of these variables), while keeping T in the field of view, and avoiding collisions. We define:

1) the *distance from the desired pose* as:

$$\rho^* = \sqrt{({}^rY_t - {}^rY_t^*)^2 + ({}^rX_t - {}^rX_t^*)^2}, \qquad (1)$$

2) the *heading to the desired position*, $\forall ({}^rX_t, {}^rY_t) \neq ({}^rX_t^*, {}^rY_t^*)$ as:

$$\alpha^* = \text{atan2}\left({}^rY_t - {}^rY_t^*, {}^rX_t - {}^rX_t^*\right), \qquad (2)$$

3) the *distance between the robot and the target* as:

$$\rho_t = \sqrt{\left({}^rY_t^2 + {}^rX_t^2\right)^2}, \qquad (3)$$

4) the *heading toward the target* as:

$$\alpha_t = \begin{cases} \text{atan2}\left({}^rY_t, {}^rX_t\right) & \text{if } ({}^rX_t, {}^rY_t) \neq (0,0), \\ 0 & \text{otherwise.} \end{cases} \qquad (4)$$

B. Obstacle Representation

For obstacle modeling, we use the occupancy grid shown in Fig. 1: it is linked to $\mathscr{F}_{\mathscr{R}}$, with cell sides parallel to X and Y. Its longitudinal and lateral extensions are limited ($X_m \leq X \leq X_M$ and $Y_m \leq Y \leq Y_M$), to ignore obstacles that are too far to jeopardize the robot. Any grid cell $\mathbf{c}$ centered at (X,Y) is considered occupied if an obstacle has been sensed in $\mathbf{c}$. The set of occupied cells with their estimated velocities, is denoted by $\mathscr{O}$:

$$\mathscr{O} = \{\mathbf{c}_1, \ldots, \mathbf{c}_n\}.$$

This is used, along with the robot geometric and kinematic characteristics, to derive possible future collisions. Indeed, the estimations of the obstacles positions and velocities are updated at every iteration, by the observer proposed in [17]. Then, for each $\mathbf{c}_i$ that may be occupied by an obstacle within time horizon T, we can predict initial $t_{i0}(\mathbf{c}_i, \mathscr{O}) \in [0,T]$ and final $t_{if}(\mathbf{c}_i, \mathscr{O}) \in [t_{i0}, T]$ obstacle occupation times, as a function of $\mathscr{O}$.

C. Tentacles

We hereby present a generalization of the tentacles-based approach proposed in [17] and [13], to omnidirectional robots. We use a set of drivable paths (tentacles) both for perception and motion execution. Each tentacle j is a semicircle that starts in R and is tangent to the robot linear velocity v. In contrast with the tentacles originally proposed in [17], our omnidirectional tentacles are characterized not only by their curvature (i.e., inverse radius) $\kappa_j = \omega/v$, but also by their course angle $\alpha_j = \text{atan2}(v_Y, v_X)$. In fact, note that on traditional non-holonomic systems, where v is aligned with the robot heading, since $v_Y = 0$, α is null (all tentacles are tangent to the robot heading). Curvature κ_j belongs to $\mathscr{K}$, a uniformly sampled set:

$$\kappa_j \in \mathscr{K} = \{-\kappa_M, \ldots, 0, \ldots, \kappa_M\}$$

and α_j belongs to $\mathscr{A}$, another uniformly sampled set:

$$\alpha_j \in \mathscr{A} = \{\alpha_{Min}, \ldots, \alpha_{Max}\} \subseteq [-\pi, \pi].$$

The total number of tentacles is the product of the number of possible curvatures by the number of course angles. An example with 36 tentacles is shown in Fig. 1. Since our tentacles are used both for perception and motion execution, a compromise between computational cost and control accuracy must be reached to tune the size of $\mathscr{K}$ and $\mathscr{A}$, i.e., their sampling intervals. Each

tentacle j is characterized by two classification areas (*collision* and *dangerous*), which are obtained by rigidly displacing, along the tentacle, two rectangular boxes, with increasing size. The boxes are overestimated with respect to the real robot dimensions. For each tentacle j, the sets of cells belonging to the two classification areas are noted $\mathscr{C}_j$ and $\mathscr{D}_j$[1]. The sets $\mathscr{O}$, $\mathscr{C}_j$ and $\mathscr{D}_j$ are used to calculate the variables required in our control law, as will be explained just below. In particular, the largest classification area $\mathscr{D}$ is used to select the safest tentacle and its risk function, while the thinnest one $\mathscr{C}$ determines the - eventually needed - deceleration.

D. Robot occupation times

For each dangerous cell in tentacle j (i.e., for each cell $\mathbf{c}_i \in \mathscr{D}_j$), we compute the *robot occupation time* t_{ij}. This is an estimate of the time at which the large box will enter the cell, assuming the robot follows the tentacle at the current velocity. To calculate $t_{ij}(\mathbf{c}_i, v, \alpha_j, \kappa_j)$, we assume that the robot motion is uniform, and displace the box at the current robot velocities, $v_X = v\cos\alpha_j$ and $v_Y = v\sin\alpha_j$, and $\omega_j = \kappa_j v$.

E. Dangerous and collision instants

Once the obstacle and robot occupation times have been calculated for each cell, we can derive the earliest time instant at which a collision between obstacle and robot may occur on each tentacle j. By either checking all cells in $\mathscr{D}_j$, or focusing just on $\mathscr{C}_j$, we discern between *dangerous instants* and *collision instants*. These are defined respectively as:

$$t_j = \inf_{\mathbf{c}_i \in \mathscr{D}_j} \left\{ t_{ij} : t_{i0} \leq t_{ij} \leq t_{if} \right\},$$

and

$$t_j^c = \inf_{\mathbf{c}_i \in \mathscr{C}_j} \left\{ t_{ij} : t_{i0} \leq t_{ij} \leq t_{if} \right\}.$$

In both cases, we seek the earliest time at which a cell is simultaneously occupied by the obstacle and by the robot box. Assuming constant robot and obstacle velocities, these metrics give an approximation of the time that the robot can travel along the tentacle without colliding.

III. Control scheme

A. Tentacle risk function

For each tentacle j, we design a *tentacle risk function*, by using t_j and tuned thresholds $t_d > 0$ and $t_s > t_d$ (d stands for dangerous, and s for safe):

$$H_j = \begin{cases} 0 & \text{if } t_j \geq t_s \\ \frac{1}{2}\left[1 + \tanh\left(\frac{1}{t_j - t_d} + \frac{1}{t_j - t_s}\right)\right] & \text{if } t_d < t_j < t_s \\ 1 & \text{if } t_j \leq t_d. \end{cases}$$

Note that H_j smoothly varies from 0, when possible collisions are in the far future, to 1, when they are forthcoming. If $H_j = 0$, the tentacle is tagged as *clear*.

[1]For further details on the derivation of $\mathscr{C}_j$ and $\mathscr{D}_j$, refer to [13].

To determine the best behaviour to adopt (among visual target tracking and obstacle avoidance), we assess the danger of the environment via the risk function of the *best tentacle* (κ_b, α_b) (selected considering both the visual and avoidance tasks, as we will see in Section IV), i.e., $H = H_b$. Depending on the value of H, we distinguish the cases explained below.

B. Safe context

In the *safe context* ($H = 0$), no dangerous obstacle is detected on the robot path. In this case, it is desirable that the robot realizes the task of driving ${}^r\mathbf{p}_t$ to ${}^r\mathbf{p}_t^*$, while keeping as much as possible T in its field of view. Since the angular velocity ω determines both the convergence of ${}^r\theta_t$ to ${}^r\theta_t^*$ (for pose regulation) and that of α_t to 0 (for target visibility), a compromise must be reached. We weigh the two objectives, respectively with a gain $\lambda_\omega \in [0,1]$ and with its complementary $1-\lambda_\omega$. Priority is given to target visibility when the desired pose is farther than ρ_t; then, we prioritize pose regulation. For instance, we can apply:

$$\lambda_\omega(\rho^*) = \begin{cases} 1 & \text{if} \rho^* > 2\rho_t \\ \frac{\rho^*}{2\rho_t} & \text{otherwise.} \end{cases} \tag{5}$$

The translation velocity must be aligned with the heading towards α^*, while its norm must be reduced, as the target is approached. We specify this by setting:

$$v_s(\rho^*) = \begin{cases} V & \text{if} \rho^* > \rho_v \\ \frac{\rho^*}{\rho_v} V & \text{otherwise,} \end{cases} \tag{6}$$

with $V > 0$ the maximum desired value for v_s and ρ_v the distance at which the robot should slow down. All these parameters are hand-tuned variables. Then, the specifications in the safe context are:

$$\begin{cases} v_X = v_s \ \cos\alpha^* \\ v_Y = v_s \ \sin\alpha^* \\ \omega = (1-\lambda_\omega)({}^r\theta_t - {}^r\theta_t^*) + \lambda_\omega \ \alpha_t. \end{cases} \tag{7}$$

C. Unsafe context

In the *unsafe context* ($H = 1$), dangerous obstacles are detected. The robot should circumnavigate them by following the best tentacle. This path variation can drive the target out of the camera field of view. Correspondingly, the heading must be controlled to maintain visibility of the target, just like in the safe task. The translational velocity must be reduced for safety reasons; we specify this by using a function $v_u \in [0, v_s]$ that is designed as:

$$v_u = \begin{cases} v_s & \text{if } t_b^c \geq t_s^c \\ v_s \ \sqrt{t_b^c - t_d^c / t_s^c - t_d^c} & \text{if } t_d^c < t_b^c < t_s^c \\ 0 & \text{if } t_b^c \leq t_d^c \end{cases} \tag{8}$$

(with $t_d^c > 0$ and $t_s^c > t_d^c$ two thresholds corresponding to dangerous and safe collision times) to guarantee that the vehicle decelerates (and eventually stops) as the collision instant on the best tentacle t_b^c decreases. Then, the specifications in the unsafe context are:

$$\begin{cases} v_X = v_u \ \cos\alpha_b \\ v_Y = v_u \ \sin\alpha_b \\ \omega = v_u \ \kappa_b, \end{cases} \tag{9}$$

so that the translational and angular velocities guarantee that the robot follows the best tentacle, i.e., (κ_b, α_b).

D. General control law

In *intermediate contexts* $(0 < H < 1)$, the robot should navigate between the visual path, and the best tentacle. The transition between these two extremes will be driven by H. Using all the variables above, we can write our controller for visual navigation with obstacle avoidance:

$$\begin{cases} v_X = (1-H)\ v_s\ \cos\alpha^* + H\ v_u\ \cos\alpha_b \\ v_Y = (1-H)\ v_s\ \sin\alpha^* + H\ v_u\ \sin\alpha_b \\ \omega = (1-H)\left((1-\lambda_\omega)\left({}^r\theta_t - {}^r\theta_t^*\right) + \lambda_\omega\ \alpha_t\right) + H\ v_u\ \kappa_b. \end{cases} \tag{10}$$

In the next section, we explain the stategy that is adopted to select the best tentacle.

IV. TENTACLES SELECTION STRATEGY

A. Sorting tentacles

To calculate the best tentacle, we must define a criterion for sorting tentacles, in order to assess their proximity in terms of control effort. The sorting will be needed to explore the whole set to find (κ_b, α_b). Considering that varying the robot position is more energy-consuming than varying its orientation (the robot mass being more important than its moment of inertia), we use position alone to sort the tentacles. In practice, we sort all tentacles based on the future robot position after time Δt (iteration duration). We define a fixed frame $\mathscr{F}_\mathcal{O}(O, X_O, Y_O)$ (See Fig. 3) that represents the robot frame at the beginning of an iteration. For each tentacle j, we must then calculate the robot position ${}^o\mathbf{p}_r = [{}^oX_r\ {}^oY_r]^\top$ in $\mathscr{F}_\mathcal{O}$ after the robot has followed the tentacle for time Δt. We have the following relationship between the robot pose derivative and the control inputs:

$$\begin{cases} {}^o\dot{\theta}_r = \omega_j \\ {}^o\dot{X}_r = v_X\ \cos({}^o\theta_r) - v_Y\ \sin({}^o\theta_r) \\ {}^o\dot{Y}_r = v_X\ \sin({}^o\theta_r) + v_Y\ \cos({}^o\theta_r). \end{cases} \tag{11}$$

Rewriting the control inputs in function of κ_j and α_j:

$$\begin{cases} \omega_j = v\ \kappa_j \\ v_X = v\ \cos\alpha_j \\ v_Y = v\ \sin\alpha_j, \end{cases} \tag{12}$$

with $v = \sqrt{v_X^2 + v_Y^2}$ the current robot linear velocity. From (11) and (12), we can rewrite the robot pose derivative in function of κ_j and α_j :

$$\begin{cases} {}^o\dot{\theta}_r = v\ \kappa_j \\ {}^o\dot{X}_r = v\ (\cos\alpha_j\ \cos\ ({}^o\theta_r) - \sin\alpha_j\ \sin({}^o\theta_r)) \\ {}^o\dot{Y}_r = v\ (\cos\alpha_j\ \sin({}^o\theta_r) + \sin\alpha_j\ \cos({}^o\theta_r)) \end{cases}, \tag{13}$$

By integrating this expression, we can obtain the robot position at Δt. We distinguish two cases according to κ_j:

- if $\kappa_j = 0$, the robot position is defined as :

$$\begin{cases} {}^oX_r = v\ \Delta t\ \cos\alpha_j \\ {}^oY_r = v\ \Delta t\ \sin\alpha_j, \end{cases} \tag{14}$$

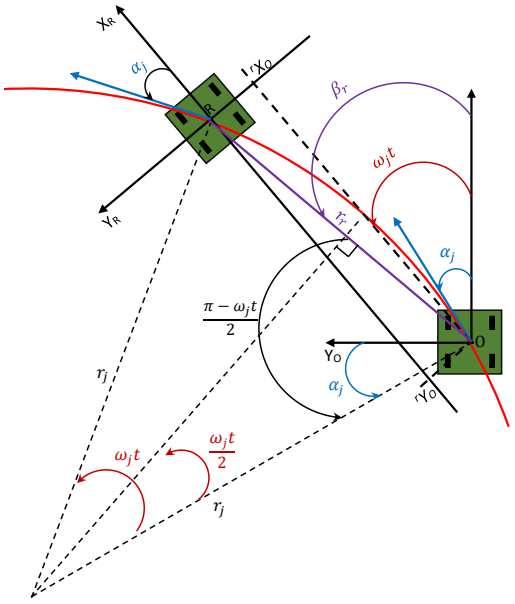

Fig. 3. Variables needed for tentacle selection.

- if $\kappa_j \neq 0$ the robot position is given by:

$$\begin{cases} {}^oX_r = 2/\kappa_j\ \sin(\kappa_j\ v\ \Delta t/2)\ \cos(\alpha_j + \kappa_j\ v\ \Delta t/2) \\ {}^oY_r = 2/\kappa_j\ \sin(\kappa_j\ v\ \Delta t/2)\ \sin(\alpha_j + \kappa_j\ v\ \Delta t/2). \end{cases} \tag{15}$$

Let us convert the position to Polar coordinates (r_r, β_r), with $r_r = \sqrt{{}^oX_r^2 + {}^oY_r^2}$ and $\beta_r = \text{atan2}({}^oY_r, {}^oX_r)$

$$r_r(\alpha_j, \kappa_j) = \begin{cases} v\ \Delta t & \text{if} \kappa_j = 0 \\ \frac{2}{\kappa_j}\ \sin\left(\frac{\kappa_j}{2}\ v\ \Delta t\right) & \text{otherwise,} \end{cases} \tag{16}$$

$$\beta_r(\alpha_j, \kappa_j) = \alpha_j + \kappa_j\ v\ \frac{\Delta t}{2}. \tag{17}$$

Since the robot movement is considered small during an iteration, $\sin\left(\frac{\kappa_j}{2} v \Delta t_i\right) \approx \frac{\kappa_j}{2} v \Delta t_i$, and therefore r_r will not vary much from a tentacle to the other $(r_r \approx v\Delta t_i)$. Hence, the variations of the robot position are only caused by variations of β_r. Therefore, we have decided to sort the tentacles set according to the values of β_r, calculated using (17). Accordingly, we compute, for each tentacle j, a sorting angle $\beta_{r,j}$.

B. Tentacles guaranteeing visibility

In this section, we introduce the visibility constraints in our strategy of obstacles avoidance. In fact, to ensure that the target is not lost during navigation, the best tentacle should allow the robot to circumnavigate the obstacles while keeping the target in sight. However, this is not possible for all tentacles (since the camera looks in the direction of the robot heading). In particular, we consider as tentacles guaranteeing visibility, those that maintain the target in the field of view after time Δt.

For a given tentacle j, we define (see Fig. 3):

1) $({}^oX_t, {}^oY_t)$: the target position (at $t = 0$) in the initial robot frame $\mathscr{F}_\mathcal{O}(R_O, X_O, Y_O)$

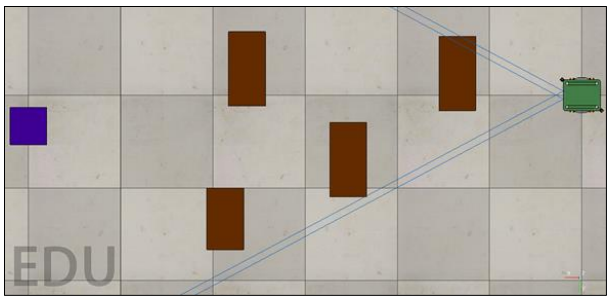

Fig. 4. V-REP simulation, with four obstacles (1 static, 3 moving).

2) $({}^rX_t, {}^rY_t)$: the target position (at $t = \Delta t$) in the robot frame $\mathscr{F}_{\mathscr{R}}(R_R, X_R, Y_R)$.

The relation between these two positions is given by :

$$\begin{bmatrix} {}^rX_t \\ {}^rY_t \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\omega_j\ \Delta t) & \sin(\omega_j\ \Delta t) & {}^rX_o \\ -\sin(\omega_j\ \Delta t) & \cos(\omega_j\ \Delta t) & {}^rY_o \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^oX_t \\ {}^oY_t \\ 1 \end{bmatrix} \tag{18}$$

where $({}^rX_o, {}^rY_o)$ are the coordinates of the origin of $\mathscr{F}_{\mathscr{O}}$ in $\mathscr{F}_{\mathscr{R}}$. Typically, after Δt these are:

$$\begin{cases} {}^rX_o = r_r\ \cos(\beta_r - \omega_j\ \Delta t + \pi) \\ {}^rY_o = r_r\ \sin(\beta_r - \omega_j\ \Delta t + \pi) \end{cases}. \tag{19}$$

We distinguish two cases:

- when $\kappa_j \neq 0$, the target position $({}^rX_t, {}^rY_t)$ in the robot frame after Δt is given by:

$$\begin{cases} {}^rX_t = \cos(\kappa_j\ v\ \Delta t)\ {}^oX_t + \sin(\kappa_j\ v\ \Delta t)\ {}^oY_t \\ \quad +2/\kappa_j\ \sin(\kappa_j\ v\ \Delta t/2)\ \cos(\alpha_j - \kappa_j\ v\ \Delta t/2 + \pi) \\ {}^rY_t = -\sin(\kappa_j\ v\ \Delta t)\ {}^oX_t + \cos(\kappa_j\ v\ \Delta t)\ {}^oY_t \\ \quad +2/\kappa_j\ \sin(\kappa_j\ v\ \Delta t/2)\ \sin(\alpha_j - \kappa_j\ v\ \Delta t/2 + \pi). \end{cases} \tag{20}$$

- If $\kappa_j = 0$, the previous expression is:

$$\begin{cases} {}^rX_t = {}^oX_t - v\ \Delta t\ \cos(\alpha_j) \\ {}^rY_t = {}^oY_t - v\ \Delta t\ \sin(\alpha_j). \end{cases} \tag{21}$$

To keep the target in the robot field of view, tentacle j has to satisfy the following constraints:

$$\begin{cases} {}^rX_t > 0 \\ -\tan(\beta/2)\ {}^rX_t < {}^rY_t < \tan(\beta/2)\ {}^rX_t, \end{cases} \tag{22}$$

with β the camera field of view (see Fig. 1). In summary, to determine wether tentacle j guarantees target visibility, we inject its (κ_j, α_j) into (20) or (21) and then check wether the resulting ${}^rX_t$, ${}^rY_t$ verify constraints (22)

C. Selecting the best tentacle (κ_b, α_b)

The best tentacle selection strategy is as follows.

1) All tentacles are sorted according to their $\beta_{r,j}$, calculated with (17).
2) All tentacles that guarantee the visibility constraints are derived as explained in Sect. IV-B.
3) We compute: the course angle $\alpha_v = \arctan(v_Y/v_X)$, curvature $\kappa_v = \omega/v_s$, hence β_r of the path that the robot would perform if there were no obstacles, i.e., if the safe context controller (7) was applied.

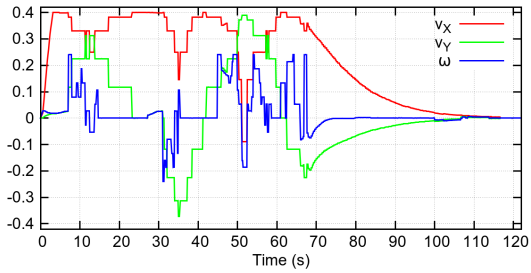

Fig. 5. V-REP youBot Control inputs during the simulation.

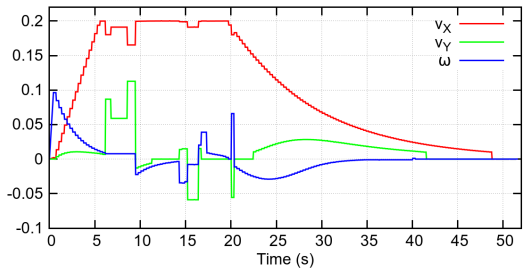

Fig. 6. MPO700 Control inputs.

4) The tentacle that best approximates the visual path (the nearest one in terms of β_r) is computed; we denote it as the *visual task tentacle* and its situation risk function as H_v.
5) If $H_v = 0$, the visual task tentacle is clear and can be followed: we set $\alpha_b = \alpha_v$ and $\kappa_b = \kappa_v$.
6) Instead, if $H_v \neq 0$, we seek a clear tentacle (i.e., one with $H_j = 0$). a) first, we search among the tentacles with $\beta_{r,j}$ between the one of the visual task tentacle and that of the best tentacle at the previous iteration, b) if many clear tentacles are present, the nearest to the visual task tentacle is chosen, c) if none of them is clear, we search among the others (the tentacles that are not between the visual task and the previous best tentacles), d) If none of the visibility guaranteeing tentacles is clear, the one with minimum H_j is chosen. Again, the best tentacle will be the clear one that is closest to the visual task tentacle.

V. EXPERIMENTS

Here, we report the simulated and real experiments that we performed to validate our approach. These are also shown in the video attached to this paper[2]. For simulations, we used V-REP[3], with the KUKA youBot, an omnidirectional mobile robot with 4 Swedish wheels. The robot is equipped with a fixed 55.8° field of view camera, and with two 270° Hokuyo lidars operating at 40 Hz. The sampling time is fixed to $\Delta t = 0.1s$. The occupancy grid is built by projecting the laser readings, using: $X_M = Y_M = 8$ m, $X_m = Y_m = -8$ m. The cells have size 20×20 cm. We use 285 tentacles, with $\kappa_M = 0.6$

[2]Also visible online at http://bit.do/dfXJG
[3]www.coppeliarobotics.com

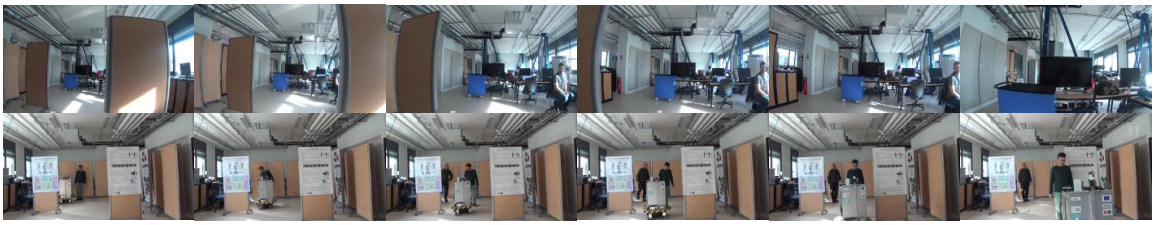

Fig. 7. Real experiment with our MPO700 platform navigating towards a blue chariot. Top: images from the on-board camera, bottom: robot behaviour while avoiding two obstacles.

m^{-1}, $\alpha_{Min} = -2\pi/3$ rad, and $\alpha_{Max} = 2\pi/3$ rad. The maximum translational velocity is set to $V = 0.4$ ms^{-1}.

Let us firstly describe the simulations, shown in Fig. 4, where the robot progresses toward the visual target (represented by a blue box) while avoiding four obstacles (represented by brown boxes) that are present in the environment. The closest obstacle is static, the second and third ones are moving in opposite direction to the robot and the farthest one is moving laterally. We have plotted, in Fig. 5, the robot control inputs during this navigation task. The plot shows that after approximately 5 s, the robot is deviated (v_y) and oriented (ω) so that the camera is pointing at the target. When the robot is near the obstacles, it starts avoiding them while keeping the target in the camera field of view. The v_x velocity increases gradually up to the maximum safe velocity $V = 0.4$ ms^{-1}. This velocity can be maintained even while avoiding obstacles, since our algorithm always selects a clear best tentacle (hence $H = 0$). Then (at $t \approx 65$ s), the environment is free again, and the visual task can be performed for the rest of the experiment. At the end, the robot decelerates and stops when the desired pose relative to the target has been reached.

After the simulations, we have also validated our approach on real experiments that have been carried out on our Neobotix MPO700 platform. This robot is an omnidirectional platform with 4 steering wheels. Note that the mobility space of this platform is reduced in comparison to the one with swedish wheels. Since the visual target detection is not yet available in our real experiments framework, we have used the robot odometry to calculate the target pose in the robot frame. The data processing and control strategy have been implemented on a laptop PC communicating with an embedded PC through a local network. The robot is controlled by a low level controller that is implemented on the embedded PC and running at a rate of 40 Hz. In this experiment, the robot navigates towards a blue chariot, with two obstacles present in the environment. The maximum safe velocity is fixed at $V = 0.2$ ms^{-1}. We show in Fig. 7 the test scenario and the images acquired by the robot camera during navigation. As can be seen, the robot has successfully maintained the target visibility while avoiding both obstacles. The control inputs are plotted in Fig. 6. Less smooth control inputs are due to the nature of our approach, that is based on sampling a set of drivable paths. However, since the sample time is higher than the one of the embedded controller, it could be possible to filter this signal at low level but this was out of scope in this work.

VI. Conclusions

In this paper, we have presented a framework that guarantees obstacle avoidance during a visual navigation task for an omni-directional mobile robot that has to deal with visibility constraints. Our technique exploits the kinematic of the platform by using omni-directional tentacles for both perception and motion execution. Simulated and real experiments show that the robot can perform the task, safely and smoothly, in spite of the visibility constraints. In the future, we plan to take into account the visual occlusions provoked by the obstacles.

References

[1] J. C. Latombe, "Robot motion planning", 1991, *Kluwer Academic, Dordredt*.

[2] S. M. LaValle, "Planning Algorithms," Cambridge, U.K.: Cambridge Univ. Press, 2006.

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots", 1985, *IEEE ICRA*.

[4] J. Borenstein and Y. Koren, "The Vector Field Histogram - Fast obstacle avoidance for mobile robots", *IEEE Trans. on Robotics and Automation*, vol. 7, no. 3, 1991, pp. 278-288.

[5] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control", *IEEE ICRA*, 1993.

[6] D. Fox, W. Burgard and S. Thrun, "The Dynamic Window approach to obstacle avoidance", in *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, 1997, pp. 23–33.

[7] J. Minguez, "The Obstacle-Restriction Method (ORM) for robot obstacle avoidance in difficult environments", *IROS*, 2005.

[8] M. Mujahad, D. Fischer, B. Mertsching and H. Jaddu "Closest Gap based (CG) reactive obstacle avoidance navigation for highly cluttered environments", *IROS*, 2010.

[9] F. Chaumette and S. Hutchinson, "Visual servo control, Part I: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13 , no. 4, pp. 82–90, December 2006.

[10] F. Chaumette and S. Hutchinson, "Visual servo control, Part II : Advanced approaches," *IEEE Robotics and Automation Magazine*, vol. 14 , no. 1, pp. 109–118, March 2007.

[11] D. Folio and V. Cadenat, "A redundancy-based scheme to perform safe vision-based tasks amidst obstacles", *IEEE Int. Conf. on Robotics and Biomimetics*, 2006, Kunming, China.

[12] M. Futterlieb, V. Cadenat, T. Sentenac, "A Navigational Framework Combining Visual Servoing and Spiral Obstacle Avoidance Techniques", ICINCO, 2014.

[13] A. Cherubini and F. Chaumette. "Visual navigation of a mobile robot with laser-based collision avoidance", *Int. Journal of Robotics Research*, vol. 32 no. 2, 2013, pp. 189–205.

[14] D. Bonnafous, S. Lacroix and T. Siméon, "Motion generation for a rover on rough terrains", *IEEE/RSJ IROS*, 2001.

[15] F. Von Hundelshausen, M. Himmelsbach, F. Hecker, A. Mueller and H.-J. Wuensche, "Driving with tentacles-Integral structures of sensing and motion", in *Journal of Field Robotics*, 2008, vol. 25, no. 9, pp. 640–673.

[16] A. Cherubini, F. Spindler and F. Chaumette, "A New Tentacles-based Technique for Avoiding Obstacles during Visual Navigation", *IEEE Int. Conf. on Robotics and Automation*, 2012.

[17] A. Cherubini, B. Grechanichenko, F. Spindler and F. Chaumette. "Avoiding moving obstacles during visual navigation", *IEEE Int. Conf. on Robotics and Automation*, 2013.