

A New Tentacles-based Technique for Avoiding Obstacles during Visual Navigation

Andrea Cherubini, Fabien Spindler, François Chaumette

▶ **To cite this version:**

Andrea Cherubini, Fabien Spindler, François Chaumette. A New Tentacles-based Technique for Avoiding Obstacles during Visual Navigation. ICRA: International Conference on Robotics and Automation, May 2012, St. Paul, Minnesota, United States. pp.4850-4855. hal-00750586

HAL Id: hal-00750586
https://inria.hal.science/hal-00750586

Submitted on 12 Nov 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A New Tentacles-based Technique for Avoiding Obstacles during Visual Navigation

Andrea Cherubini, Fabien Spindler and François Chaumette

***Abstract*— In this paper, we design and validate a new tentacle-based approach, for avoiding obstacles during appearance-based navigation with a wheeled mobile robot. In the past, we have developed a framework for safe visual navigation. The robot follows a path represented as a set of key images, and during obstacle circumnavigation, the on-board camera is actuated to maintain scene visibility. In those works, the model used for obstacle avoidance was obtained using a potential vector field. Here, a more sophisticated and efficient method, that exploits the robot kinematic model, and predicts collision at look-ahead distances, is designed and integrated in that framework. Outdoor experiments comparing the two models show that the new approach presents many advantages. Higher speeds and precision can be attained, very cluttered scenarios involving large obstacles can be successfully dealt with, and the control inputs are smoother.**

***Index Terms*— Visual Navigation, Visual Servoing, Collision Avoidance.**

I. INTRODUCTION

One of the main objectives of robotics research is the development of vehicles capable of autonomously navigating in unknown environments [1], [2]. In this field, an important task is obstacle avoidance: if possible, a collision-free trajectory to the goal should be generated; otherwise, the vehicle must brake to prevent collision [3].

The task that we focus on is outdoor visual navigation: a wheeled vehicle, equipped with an actuated pinhole camera and with a forward-looking range scanner, must follow a path represented by key images, without colliding with the ground obstacles. The camera detects the features required for navigating, while the scanner senses the obstacles (in contrast with other works, such as [4], only one sensor is used to detect the obstacles). In the past, obstacle avoidance has been integrated in visual navigation [5], [6] and path following [7], [8], by using the path geometry or the environment 3D model (including, for example, walls and doors). However, since our task is defined in the image space, we seek a merely sensor-based solution, which does not need a global model of the environment and trajectory.

In our recent work [9] we adopted a method based on potential fields [10] built on an occupancy grid. This method, however, suffered from its simplicity, which led to very conservative collision avoidance, and strongly varying control inputs. Alternative reactive strategies include: the vector field histogram [11], dynamic window [12], obstacle-restriction method [13], and closest gap [14]. The first two methods rely on a candidate set of commands; however, trap situations and oscillations may arise. In [13], these issues are solved, but, since the robot is assumed holonomic and a 3D subgoal is required, the approach is not suitable for our problem. Similarly, [14] uses the robot pose, which is noisy in our framework. Instead, we take inspiration from [15] and [16], where a set of trajectories (arcs of circles) is evaluated for navigating. However, in [15], a sophisticated probabilistic elevation map is used, and the selection of the optimal arc of circle is based on its risk and interest, which both require accurate pose estimation. Similarly, in [16] the candidate trajectories ('tentacles') are also used for free navigation, which relies on GPS way points, hence - once more - on the robot pose. A deeper comparison with this work is carried out in Sect. IV.

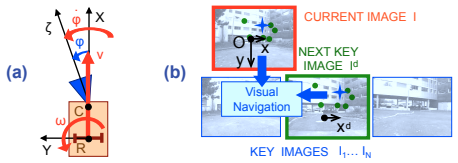


Fig. 1. General definitions. (a) Top view of the robot (orange), with actuated camera (blue). (b) Current and next key images, and key image database.

The main contribution of our work is the development of a novel, reactive, pose-independent, tentacle-inspired obstacle avoidance technique, which is perfectly suitable for appearance-based tasks, such as visual navigation. The main strengths of this approach are that it is reactive and sensor-based (it does not require any model of the environment), whilst providing useful "look-ahead" information, by exploiting the robot geometric and kinematic characteristics. Our method is experimentally validated in the framework designed in [9], and compared with the previous potential field method. In [9], we guaranteed that obstacle avoidance had no effect on the visual task, by discerning two contexts (*safe* and *unsafe*), and pursuing the corresponding tasks. Here, the tentacle-based approach intervenes both in assessing the context, and in designing the task in case of danger.

The article is organized as follows. In Sect. II, the control law from [9] is recalled. Then, two alternative obstacle models are presented: vortex potential fields (Sect. III) and tentacles (Sect. IV). Experimental results are reported in Section V, and summarized in the conclusion.

II. CONTROL SCHEME

A. General Definitions

The reader is referred to Fig. 1. We define the robot frame $\mathcal{F}_{\mathcal{R}}\,(R, X, Y)$ (R is the robot center of rotation) and image frame $\mathcal{F}_{\mathcal{I}}(O, x, y)$ (O is the image center); C is the camera optical center. The robot control inputs are:

$$\mathbf{u} = (v,\ \omega,\ \dot{\varphi})\,.$$

These are the translational and angular velocities of the vehicle, and the camera pan angular velocity. We use the normalized perspective camera model, and we assume that the sequence of images that defines the path can be tracked

A. Cherubini was with INRIA Rennes - Bretagne Atlantique, IRISA, Campus de Beaulieu 35042, Rennes, France, and is now with the Laboratory for Computer Science, Microelectronics and Robotics LIRMM - Université de Montpellier 2 CNRS, 161 Rue Ada, 34392 Montpellier, France. {Andrea.Cherubini}@lirmm.fr.
F. Spindler and F. Chaumette are with INRIA Rennes - Bretagne Atlantique, IRISA. {firstname.lastname}@inria.fr

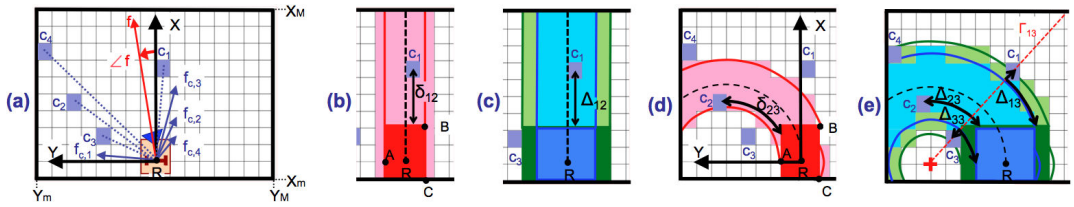


Fig. 2. Obstacle models with 4 occupied cells $c_1, \ldots, c_4$ (purple). (a) Singular cell (blue vectors) and total (red vector) vortex potential fields. (b - e) Tentacles (dashed black), with classification areas (collision in pink, dangerous central in cyan, dangerous external in green), corresponding boxes and delimiting arcs of circle (red, blue and dark green), and cell risk and collision distances (Δ_{ij}, δ_{ij}); ray of cell c_1 with respect to the third tentacle, Γ_{13} (dashed red).

with continuous $v\,(t) > 0$. This ensures safety, since only obstacles in front of the robot can be detected by our scanner.

The path that the robot must follow is represented as a database of ordered key images, such that successive pairs contain some common static visual features (points). First, the vehicle is manually driven along a *taught* path, with the camera pointing forward ($\varphi = 0$), and all the images are saved. Afterwards, a subset (database) of N key images $I_1, \ldots, I_N$ representing the path (Fig. 1(b)) is selected. Then, during autonomous navigation, the current image, noted I, is compared with the next key image $I^d \in \{I_1, \ldots, I_N\}$, and a relative pose estimation between I and I^d is used to check when the robot passes the pose where I^d was acquired. For key image selection, and visual point detection and tracking, we use the algorithm in [17]. The output of this algorithm, which is used by our controller, is the set of points visible both in I and I^d. Then, navigation consists of driving the robot forward, while I is driven to I_d. We maximize similarity between I and I^d using only the abscissa x of the centroid of the points matched on I and I^d. When I^d has been passed, the next image in the set becomes the desired one, and so on, until I_N is reached.

Along with the visual path following problem, we consider obstacles which are on the path, but not in the database, and sensed by the range scanner in a plane parallel to the ground. For obstacle modeling, we use the occupancy grid in Fig. 2(a): it is linked to $\mathcal{F}_\mathcal{R}$, with cell sides parallel to X and Y. Its extension is 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 c centered at (X, Y) is considered occupied if an obstacle has been sensed in c. For cells entirely lying in the scanner area, only the current scanner reading is considered. For all other cells, we use past readings, which are progressively displaced using odometry.

B. Control Design

The desired behaviour of the robot is related to the surrounding obstacles. When the environment is safe, the vehicle should progress forward while remaining near the taught path, with camera pointing forward ($\varphi = 0$). If avoidable obstacles are present, we apply a robot rotation for circumnavigation with an opposite camera rotation to maintain visibility. Finally, if collision is inevitable, the vehicle should simply stop. To select the behaviour, we assess the danger at time t with a *situation risk function* $H : I\!R^{*+} \mapsto [0, 1]$.

Stability of these tasks has been guaranteed in [9] by:

$$\begin{cases} v = (1 - H)\, v_s + H v_u \\ \omega = (1 - H)\, \frac{\lambda_x \left(x^d - x\right) - j_v v_s + \lambda_\varphi j_{\dot\varphi} \varphi}{j_\omega} + H \omega_u \\ \dot\varphi = H\, \frac{\lambda_x \left(x^d - x\right) - j_v v_u - j_\omega \omega_u}{j_{\dot\varphi}} - (1 - H)\, \lambda_\varphi \varphi \end{cases} \tag{1}$$

In the above equations:

- $H \in [0, 1]$ is the situation risk function introduced above; two alternative definitions, depending on the obstacle model, will be given in Sect. III and IV.
- v_s is the translational velocity in the *safe context* (i.e., when $H = 0$). It must be reduced when the features are moving quickly in the image, making tracking difficult. This is typically the case at sharp robot turns, and when the camera pan angle φ is strong. We define v_s as:

$$v_s\,(\omega, \varphi) = v_m + \frac{v_M - v_m}{4}\, \sigma \tag{2}$$

with function σ defined as:

$$\sigma : I\!R \times \left[-\tfrac{\pi}{2}, \tfrac{\pi}{2}\right] \to [0, 4]$$
$$(\omega, \varphi) \mapsto \left[1 + \tanh\left(\pi - k_\omega |\omega|\right)\right] \left[1 + \tanh\left(\pi - k_\varphi |\varphi|\right)\right].$$

Function (2) has an upper bound $v_M > 0$ (for $\varphi = \omega = 0$), and smoothly decreases to the lower bound v_m, as either $|\varphi|$ or $|\omega|$ grow. The decreasing trend of v_s is determined by empirically tuned positive parameters v_M, v_m, k_ω and k_φ. This definition of v_s yields better results than the one in [9], which was only characterized by the image x variation.

- $v_u \in [0, v_s]$ is the translational velocity in the *unsafe context* ($H = 1$). It guarantees that the vehicle slows down (and eventually stops) in dangerous situations.
- x and x_d are abscissas of the feature centroid respectively in the current and next key image.
- $\lambda_x > 0$ and $\lambda_\varphi > 0$ are empirical gains determining the convergence trend of x to x_d and of φ to 0.
- j_v, j_ω and $j_{\dot\varphi}$ are the components of the Jacobian relating $\dot x$ and the control inputs $\mathbf{u}$:

$$j_v = \frac{-\sin\varphi + x \cos\varphi}{\zeta}$$
$$j_\omega = \frac{\overline{RC}(\cos\varphi + x \sin\varphi)}{\zeta} + 1 + x^2$$
$$j_{\dot\varphi} = 1 + x^2,$$

with $\overline{RC}$ and ζ (the centroid depth, hand-tuned according to the environment characteristics) depicted in Fig. 1(a).

- ω_u is the angular velocity that makes the robot avoid collisions while advancing.

In [9], we proved that (1) is well defined by setting $\zeta > \overline{RC}/2$, and that $\forall H \in [0, 1]$ obstacle avoidance has no effect on the visual task. Our main contribution here will be in the obstacle model, i.e., in the definition of the variables H, v_u and ω_u according to the danger. Two alternative models will be presented and confronted experimentally. To carry out the comparison, the vortex potentials [9] are briefly recalled in Sect. III. Then, our novel tentacle-based approach is designed and presented in Sect. IV.

III. VORTEX POTENTIAL FIELDS

This approach is illustrated in Fig. 2(a). Given an arbitrary integer K, obstacles are modeled using the latest $2K+1$ scans. For each cell $c=(X,Y)$, we define the $2K+1$ *occupancies* r at the j-th oldest iteration as:

$$r_j(c)=\{0,1\}, \qquad j=0,\dots,2K+1.$$

We set $r_j=1$ if an obstacle has been sensed in c at the j-th iteration prior to the current one, and 0 otherwise. Then, we associate to each cell a coefficient $\mu(c)$, obtained by linear combination of the occupancies, weighted with a normalized Gaussian filter that smoothens the cell effect over time:

$$\mu(c)=\sum_{j=0}^{2K+1}\frac{e^{-(j-K)^2/K}}{\sqrt{K\pi}}\ r_j(c).$$

The maximum weight is at the K-th latest scan, to avoid overshoot at a new obstacle detection. If the robot velocity is negligible with respect to the scanner frequency, and K is small, the effect of motion on the occupancies can be ignored. In [9], we showed that these assumptions are appropriate in our setup.

The potential associated to each cell $c\neq R$, is defined as:

$$U_c=\frac{\mu(c)}{\|c\|},$$

where $\|c\|$ is the distance from R to c. We define the vortex field for each cell as the rotor of U_c:

$$\mathbf{f}_c=\begin{bmatrix} f_{c,X}\\ f_{c,Y}\end{bmatrix}=\begin{bmatrix} \pm\dfrac{\partial U_c}{\partial Y}\\ \mp\dfrac{\partial U_c}{\partial X}\end{bmatrix}=\mu(c)\begin{bmatrix} \mp\dfrac{Y}{\|c\|^3}\\ \pm\dfrac{X}{\|c\|^3}\end{bmatrix}.$$

The signs of $f_{c,X}$ and $f_{c,Y}$ depend on X, so that the field always points forward, and the fields $f_{c,i}$ generated by all cells are superimposed to obtain the total field:

$$\mathbf{f}=\sum_i \mathbf{f}_{c,i}.$$

We then use the magnitude and phase of $\mathbf{f}$ (denoted $|\mathbf{f}|$ and $\angle\mathbf{f}$), and two tuned thresholds Δ_d and Δ_s such that $0<\Delta_d<\Delta_s$, to design the situation risk function as:

$$H=\begin{cases}0 & \text{if } |\mathbf{f}|=0\\ k_f|\angle\mathbf{f}| & \text{if } \frac{1}{|\mathbf{f}|}\geq\Delta_s\\ 1 & \text{if } \frac{1}{|\mathbf{f}|}\leq\Delta_d\\ \frac{1+k_f|\angle\mathbf{f}|}{2}+\frac{1-k_f|\angle\mathbf{f}|}{2}\tanh\left(\frac{\Delta_d}{1-|\mathbf{f}|\Delta_d}+\frac{\Delta_s}{1-|\mathbf{f}|\Delta_s}\right) & \text{otherwise.}\end{cases}$$

Note that $H=0$ if no obstacle is detected, and it is bounded by 1. For small $|\mathbf{f}|$, H is determined by $\angle\mathbf{f}$: the obstacles are far enough to be circumnavigated, and parameter $k_f\in\left]\,0,\ \frac{2}{\pi}\,\right]$ weighs the danger provoked by the field orientation. Instead, for large $|\mathbf{f}|$, the obstacles are 'too near', thus $H=1$. A hyperbolic function is used to interpolate in between.

In the presence of dangerous obstacles (i.e., for large H), the robot should slow down, and eventually stop. Imposing $v=0$ when $H=1$ in (1), yields:

$$v_u=0.$$

Hence, in the general case, the first equation in (1) becomes:

$$v=(1-H)\,v_s.$$

The velocity is reduced from v_s to 0, as the risk increases.

To ensure that to circumnavigate the obstacles in dangerous contexts, the robot aligns its heading with $\mathbf{f}$, we use a gain $\lambda_f>0$, and set the desired unsafe angular velocity to:

$$\omega_u=\lambda_f\angle\mathbf{f}.$$

IV. TENTACLES

A. Related work

Our new technique is mainly inspired from [16], although some differences, listed below, have been introduced, to deal with the specific constraints of appearance-based navigation.

The design in [16], although strengthened by a dynamic analysis, is justified only empirically. For example, the danger of the cells is associated to their orthogonal projection on the arc of circle, without measuring the actual distance to collision. Instead, in the case of two cells with same orthogonal projection (e.g., c_1 and c_3 in Fig. 2(e)), the internal cell should be considered more dangerous. Besides, in [16], the requested trajectory is defined by GPS way points. Hence, in the absence of GPS, the approach in [16] becomes purely reactive, while our approach can still follow a path, defined by the image database.

In contrast with that work, our approach does not require a 3D notion of the goal. It must be purely reactive, since the visual task and the obstacle avoidance task are defined in different state spaces (respectively in the image and in the local planar surroundings). As in [16], our grid is built locally at each new scanner acquisition, without accumulating past data, and each tentacle is associated to some classification areas overestimating the vehicle encumbrance. However, these areas are defined and used differently. Their definition is associated to the rigid body kinematics of the boxes representing the vehicle encumbrances, to consider distances to collision. Then, with the largest areas we select the safest tentacle and its risk, and with the thinnest one, the eventual deceleration. Finally, since our vehicle navigates at slightly varying speeds, we do not relate the tentacle sets to v (as in [16]), so that a reduced number of candidate paths is sufficient. Our approach operates locally and instantaneously, without planning nor deriving the robot pose, to determine the values of H, v_u and ω_u in (1).

B. Classification areas and metrics

We use, along with the set of all *occupied grid cells*:

$$\mathcal{O}=\{c_1,\dots,c_n\},$$

a set of drivable paths (tentacles). Each tentacle j is a semi-circle that starts in R, is tangent to X, and is characterized by its curvature (i.e., inverse radius) κ_j, which belongs to $\mathcal{K}$, a uniformly sampled set:

$$\kappa_j\in\mathcal{K}=\{-\kappa_M,\dots,\kappa_M\}.$$

The maximum desired curvature $\kappa_M>0$, must be feasible considering the robot kinematics. We consider an odd number of tentacles, so that a straight tentacle (with $\kappa=0$) also exists. To illustrate our method, in Fig. 2(b-e), the straight and the sharpest counterclockwise ($\kappa=\kappa_M$) tentacle are shown in dashed black. When a total of 3 tentacles is used, these correspond respectively to $j=2$ and $j=3$.

Each tentacle j is characterized by three classification areas (*collision*, *dangerous central*, and *dangerous external*), obtained by rigidly displacing, along the tentacle, three rectangular boxes (red, blue, and dark green in Fig. 2), with increasing width. The boxes, with same height and different widths, are all overestimated with respect to the robot dimensions, to ensure that, in the presence of disturbances, the actual path is included in the classification area. All three areas are delimited by the box and by three arcs of circle (or lines, in the particular case $\kappa = 0$) concentric with the tentacle, and starting from three points (denoted A, B and C) on the box perimeter. In all cases, these points are: the two corners on one side of the box (the outer side must be used if $\kappa \neq 0$), and the intersection between the other side (intern if $\kappa \neq 0$) and the rear wheel axis. For the collision areas, A, B and C are shown in Fig. 2. We then associate each area to the set of all cells (pink, cyan and light green in Fig. 2) whose center lies within the area. For tentacle j, the three sets of cells are noted $\mathcal{C}_j$, $\mathcal{D}_j$ and $\mathcal{E}_j$. The only exception is with cells belonging to the *dangerous central* set, which are not considered in the *external* set as well: $\mathcal{D}_j \bigcap \mathcal{E}_j = \emptyset$.

During navigation, $\mathcal{O}$ is used, along with the sets just defined, to calculate a candidate risk function $H_j \in [0, 1]$ for each tentacle j, and select the best tentacle accordingly. Then, the unsafe translational velocity on the best tentacle, v_u, is calculated to adapt the speed to the potential danger, and to finally derive ω_u. All these steps are detailed below.

C. Tentacle risk function

The tentacle risk function H_j is derived from the risk distance of all occupied cells in the *dangerous* areas. This distance is denoted $\Delta_{ij} \geq 0$ for each $c_i \in \mathcal{O} \bigcap (\mathcal{D}_j \bigcup \mathcal{E}_j)$.

For occupied cells in the central set $\mathcal{D}_j$, Δ_{ij} is the distance that the middle boundary box (blue) would cover along tentacle j before touching the cell center.

For the external set, we consider only the subset $\bar{\mathcal{E}}_j \subseteq \mathcal{O} \bigcap \mathcal{E}_j$ of cells which reduce the clearance in the tentacle normal direction. For each external occupied cell, we denote Γ_{ij} the ray starting at the tentacle center and passing through c_i. Cell c_i is added to $\bar{\mathcal{E}}_j$ if and only if, in $\mathcal{D}_j \bigcup \mathcal{E}_j$, there is at least an occupied cell crossed by Γ_{ij} on the other side of the tentacle. In the example of Fig. 2(e), $\mathcal{O} \bigcap \mathcal{E}_3 = \{c_1, c_3, c_4\}$, whereas $\bar{\mathcal{E}}_3 = \{c_1, c_3\}$. Then, for cells in $\bar{\mathcal{E}}_j$, Δ_{ij} is the sum of two terms: the distance from the center of c_i to its normal projection on the perimeter of the dangerous central area, and the distance that the middle boundary box would cover along tentacle j before reaching the normal projection. The derivation of Δ_{ij} is illustrated, in Fig. 2, for 4 cells. Note that for a given cell, Δ_{ij} may have different values (or even be undefined) according to the tentacle.

When all risk distances on the tentacle are calculated, we compute Δ_j as their minimum:

$$\Delta_j = \inf_{c_i \in (\mathcal{O} \cap \mathcal{D}_j) \cup \bar{\mathcal{E}}_j} \Delta_{ij}.$$

If $(\mathcal{O} \bigcap \mathcal{D}_j) \bigcup \bar{\mathcal{E}}_j \equiv \emptyset$, $\Delta_j = \infty$. In the example of Fig. 2, $\Delta_2 = \Delta_{12}$ and $\Delta_3 = \Delta_{33}$. Obviously, overestimating the bounding box sizes leads to more conservative Δ_j.

We then use Δ_j and two tuned thresholds Δ_d and Δ_s ($0 < \Delta_d < \Delta_s$), to design the tentacle risk function:

$$H_j = \begin{cases} 0 & \text{if } \Delta_j \geq \Delta_s \\ \frac{1}{2}\left[1 + \tanh\left(\frac{1}{\Delta_j - \Delta_d} + \frac{1}{\Delta_j - \Delta_s}\right)\right] & \text{if } \Delta_d < \Delta_j < \Delta_s \\ 1 & \text{if } \Delta_j \leq \Delta_d. \end{cases} \quad (3)$$

Note that H_j smoothly varies from 0, when the dangerous cells on the tentacle (if any) are far, to 1, when they are near. If $H_j = 0$, the tentacle is tagged as *clear*. All the H_js are compared (with the strategy explained below), to determine H in (1) and select the *best tentacle* for navigation.

D. Situation risk function and best tentacle

Here we detail our strategy for determining the best tentacle curvature κ_b for navigation, and therefore $\mathbf{u}$ in (1).

Initially, we calculate the path curvature $\kappa = \omega / v \in I\!R$ that the robot would follow if there were no obstacles. Replacing $H = 0$ in (1), it is:

$$\kappa = \left[\lambda_x \left(x^d - x\right) - j_v v_s + \lambda_\varphi j_{\dot{\varphi}} \varphi\right] / j_\omega v_s,$$

which is always well-defined, since $j_\omega \neq 0$ and we have set $v_s > v_m > 0$. We obviously constrain κ to the interval of feasible curvatures $[-\kappa_M, \kappa_M]$. Then, we derive the two neighbors of κ among all the existing tentacle curvatures:

$$\kappa_n, \kappa_{nn} \in \mathcal{K} \text{ such that } \kappa \in [\kappa_n, \kappa_{nn}).$$

Let κ_n be the nearest one, i.e., the curvature of the tentacle that best approximates the safe path[1]. We denote it as the *visual task tentacle*. The situation risk function H_v of that tentacle, which measures the risk on the visual path, is obtained by linear interpolation of its neighbours:

$$H_v = \frac{(H_{nn} - H_n)\,\kappa + H_n \kappa_{nn} - H_{nn} \kappa_n}{\kappa_{nn} - \kappa_n}. \quad (4)$$

If $H_v = 0$, the visual task tentacle can be followed: we set $\kappa_b = \kappa_n$, and we apply (1) with $H = 0$. Instead, if $H_v \neq 0$, we seek a clear tentacle ($H_j = 0$). First, to avoid abrupt control changes, we only search among the tentacles between the visual task one and the best one at the previous iteration[2], noted κ_{pb}. If many clear ones are present, the closest to the visual task tentacle is chosen. If none of the tentacles with curvature in $[\kappa_n, \kappa_{pb}]$ is clear, we search among the others. Again, the best tentacle will be the clear one that is closest to κ_n and, in case of ambiguity, the one closest to κ_{nn}. If a clear tentacle has been found, we select it and set $H = 0$. Instead, if no tentacle in $\mathcal{K}$ is clear, the one with minimum H_j calculated using (3) is chosen, and H is set equal to that H_j. Ambiguities are again solved first with the distance from κ_n, then from κ_{nn}.

In all cases, the unsafe translational velocity v_u is derived from the obstacles on the best tentacle, as explained below.

E. Unsafe translational velocity

The unsafe translational velocity is derived from the collision distance on the best tentacle, δ_b, which is a conservative approximation of the maximum distance that the robot can travel along the best tentacle without colliding. Since the thinner (red) box contains the robot, if R follows the best tentacle, collisions can only occur in occupied cells in $\mathcal{C}_b$.

[1] We consider that intervals are defined even when the first endpoint is greater than the second: $[\kappa_n, \kappa_{nn})$ must be read $(\kappa_{nn}, \kappa_n]$ if $\kappa_n > \kappa_{nn}$.

[2] At the first iteration, we set $\kappa_{pb} = \kappa_n$.

In fact, the collision with cell c_i will occur at the distance, denoted $\delta_{ib} \geq 0$, that the red box would cover along the best tentacle, before touching the center of c_i. The derivation of δ_{ib} is illustrated in Fig. 2 for four occupied cells.

Then, we define δ_b as the minimum among the collision distances of all occupied cells in $\mathcal{C}_b$:

$$\delta_b = \inf_{c_i \in \mathcal{O} \cap \mathcal{C}_b} \delta_{ib}.$$

If all cells in $\mathcal{C}_b$ are free, $\delta_b = \infty$. In the example of Fig. 2, assuming the best tentacle is the straight one ($b = 2$), $\delta_b = \delta_{12}$. Again, oversizing the box leads to more conservative δ_b.

The translational velocity must be designed accordingly. Let δ_d and δ_s be two tuned thresholds such that $0 < \delta_d < \delta_s$. If the probable collision is far enough ($\delta_b \geq \delta_s$), v can be maintained at the safe value defined in (2). Instead, if it is near ($\delta_b \leq \delta_d$), the robot should stop. To comply with the boundary conditions $v_u(\delta_d) = 0$ and $v_u(\delta_s) = v_s$, in between we apply a constant deceleration:

$$a = v_s^2 / 2(\delta_d - \delta_s) < 0.$$

Since the distance required for braking at velocity $v_u(\delta_b)$ is:

$$\delta_b - \delta_d = -v_u^2 / 2a,$$

the expression of the unsafe translational velocity becomes:

$$v_u(\delta_b) = \begin{cases} v_s & \text{if } \delta_b \geq \delta_s \\ v_s \sqrt{\delta_b - \delta_d / \delta_s - \delta_d} & \text{if } \delta_d < \delta_b < \delta_s \\ 0 & \text{if } \delta_b \leq \delta_d, \end{cases} \quad (5)$$

in order to decelerate as the collision distance decreases.

F. Unsafe angular velocity

Once the best tentacle (with curvature κ_b) is chosen, and the corresponding v_u is obtained with (5), we set:

$$\omega_u = v_u \kappa_b. \quad (6)$$

Setting this value of unsafe angular velocity in (1) guarantees that when $H = 1$, the robot precisely follows the best tentacle, with translational velocity v_u.

G. Discussion

The values of H, v_u and ω_u derived as explained above are inserted in (1) to derive the control inputs $\mathbf{u}$.

When $H = 0$, the robot tracks at its best the taught path: the image error is regulated by ω, while v is set to v_s to improve tracking, and the camera is driven forward ($\varphi = 0$). This occurs if the 2 neighbour tentacles are clear, while in the vortex approach, even a single occupied cell would generate $H > 0$. Thus, one advantage of the new approach is that only obstacles on the visual path are taken into account. When $H = 1$, $\dot{\varphi}$ ensures the visual task, and the two other inputs guarantee that the best tentacle is followed: $\omega / v = \kappa_b$. The applied curvature fluctuates less than with the potential fields, where it is driven by strongly varying $\mathbf{f}$.

In general, the robot navigates between the taught path, and the best path considering obstacles. Only the transition, but not the speed, is driven by H. In fact, note that, for all $H \in [0, 1]$, when $\delta_b \geq \delta_s$: $v = v_s$. This is another advantage of tentacles: a high velocity can be applied if the path is clear up to δ_s. With vortex fields, instead, the vehicle stops frequently (as soon as $H = 1$). Finally, one may object that processing can be costly. However, since Δ_{ij} and δ_{ij} are invariant geometric characteristics related to the cell positions, their values at each cell can be computed and stored offline, for use when the cell becomes occupied.

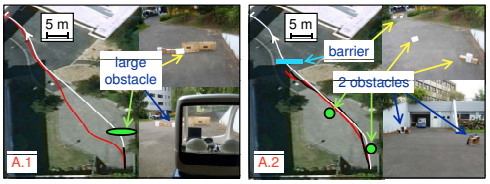


Fig. 3. Two obstacle scenarios with taught (white) and replayed (red: using tentacles, black: using potentials) paths.

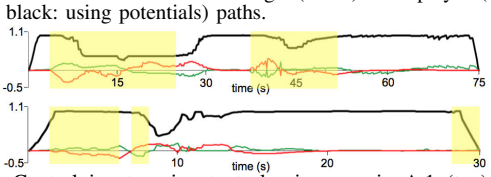


Fig. 4. Control inputs using tentacles in scenario A.1 (top) and A.2 (bottom): v (black, in ms^{-1}), ω (green, in rads^{-1}), and $\dot{\varphi}$ (red, in rads^{-1}). The iterations with strong H are highlighted in yellow.

V. EXPERIMENTS

Here, we report the experiments (also shown in the video attached to this paper) that we performed to validate our approach. All experiments have been carried out on our CyCab robot, equipped with a 70° field of view, B&W Marlin (F-131B) camera mounted on a TRACLabs Biclops Pan/Tilt head, and with a 2-layer, 110° scanning angle, laser SICK LD-MRS. The grid is built by projecting the readings from the 2 layers on the ground, and by using: $X_M = Y_M = 10$ m, $X_m = -2$ m, $Y_m = -10$ m. The cells have size 20×20 cm. We use 21 tentacles, with $\kappa_M = 0.35$ m^{-1} (the Cycab maximum applicable curvature).

First, we have compared the two obstacle avoidance methods (vortex potential fields and tentacles) in the experiments in Fig. 3. The taught path (denoted A, and white in the figure) is 60 m long. By placing various obstacles, we design the scenarios A.1 (with a long obstacle perpendicular to the path) and A.2 (with two avoidable obstacles and a blocking barrier). Then, we attempt to replay path A in each scenario, using either technique. The replayed paths, estimated from odometry and snapshots, are drawn in black (potentials) and red (tentacles). The control inputs $\mathbf{u}$ using tentacles are plotted in Fig. 4, with dangerous iterations (i.e., with strong H) highlighted in yellow. The smooth trend of $\mathbf{u}$ at the beginning and end is due to the acceleration saturation carried out by the CyCab low-level control. In Fig. 5, we have plotted the curvatures ω / v that are applied in the 4 experiments. Since our preliminary tests, it was clear that tentacles could be implemented at higher speeds than potentials, since they provide look-ahead information, and a stabler model. Hence, in (2), we set: $v_M = 1$ ms^{-1} with tentacles, $v_M = 0.4$ with potentials, and $v_m = 0.3$ for both. For $v_M > 0.4$, potentials fail to avoid the obstacles, while with tentacles, the velocity has been limited to reduce the motion of features between successive images; the maximum speed attainable by the CyCab is 1.3 ms^{-1} anyway.

In scenario A.1, the long obstacle cannot be circumnavigated using potentials. It induces oscillations on the orientation $\angle \mathbf{f}$, hence on the applied curvature (solid black curve in Fig. 5), until the robot is too near to the obstacle, and eventually stops. Instead, using tentacles, the obstacle is overtaken on the left, while the camera rotates right to maintain scene visibility (green and red curves in Fig. 4, top). The robot successfully reaches the final key image and completes navigation, although it is driven over 5 meters away from the taught path. In practice, soon after the obstacle

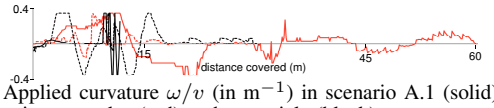


Fig. 5. Applied curvature ω/v (in m^{-1}) in scenario A.1 (solid) and A.2 (dashed) using tentacles (red) and potentials (black).

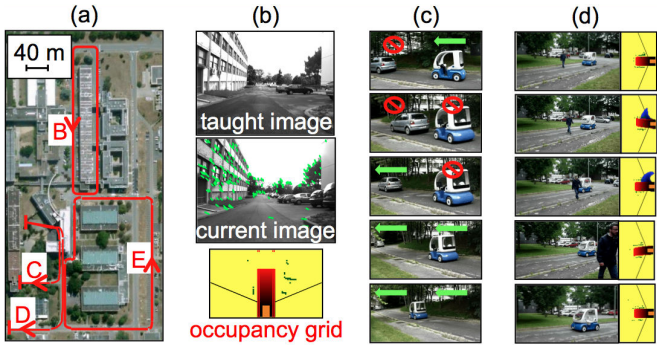


Fig. 6. Map of the four navigation paths B, C, D, E (a), and validation with: irrelevant obstacles (b), traffic (c) and a moving pedestrian (d). The visual task tentacle is shown in red, and the best tentacle (when different) in blue; only cells that can activate H (i.e., cells at distance $\Delta < \Delta_s$) have been drawn, with brightness proportional to Δ.

is detected, tentacles with first positive ($5-16$ s), and then negative ($16-25$ s) curvatures are selected. Since these tentacles are clear, v is reduced only for visual tracking, by (2) (black curve in Fig. 4, top). This is an advantage over the too conservative potential fields, which do not consider the real collision risk, and force the stop. After 25 s, the environment returns clear ($H = 0$), so the visual tentacle can be followed again, and the robot is driven back to the path. Then ($38-52$ s) a small bush on the left triggers H and causes a rotation along with a slight decrease in v. Then the context returns safe, and the visual path can be followed until the end. The translational velocity averaged over the experiment is 0.79 ms^{-1}, which is more than twice the speed reached in [9].

In scenario A.2, with both methods Cycab navigates without colliding and stops at the barrier. However, as aforementioned, the average navigation velocities v are very different, i.e., 0.83 ms^{-1} with tentacles, and only 0.35 ms^{-1} with potentials. Moreover, the applied curvature varies less in the first case (see dashed curves in Fig. 5). This leads to smoother and faster navigation than with potentials. When the barrier is reached, the vortex method behaves as in A.1 to force the stop. Instead, the new method seeks a feasible path until all tentacles are occupied. Then, (5) makes v gradually decrease to zero (black curve in Fig. 4, bottom).

After these results, which confirmed the advantages of the new over the old method, we have run some experiments, uniquely with tentacles, on longer and more crowded paths, shown in Fig. 6(a). The Cycab completed all paths (including 650 m long path E), while dealing with various natural and unpredictable obstacles, such as parked and driving cars, pedestrians, and even bicycles. A first result that emerged from the experiments was that, by assessing the collision risk only along the visual path, non-dangerous obstacles (e.g., walls or cars parked on the sides) are not taken into account. This is clear from Fig. 6(b): the cars parked on the right (which were not present during teaching) do not belong to any of the visual task tentacle classification areas. Hence, they are irrelevant, and do not deviate the robot from the path. Another nice behavior is shown in Fig. 6(c): if a stationing car is unavoidable, the robot decelerates and stops with (5), but, as soon as the car departs, it gradually accelerates (again with (5)), to resume navigation. An experiment with a crossing pedestrian is presented in Fig. 6(d). The pedestrian is considered irrelevant, until it enters the visual task tentacle. Then, the clockwise tentacles (blue in the figure) are selected to pass between the person and the right sidewalk. When the path is clear again, the robot turns left to recover it.

VI. CONCLUSIONS

We have presented a novel, robust and reactive technique for avoiding obstacles with a wheeled robot. By exploiting the robot kinematics, we can predict collisions at look-ahead distances along candidate circular paths (tentacles). Since our method is sensor-based and pose-independent, it is perfectly suited for visual navigation. Extensive experiments show that it generates smoother control inputs than its predecessor, and that it can be applied in realistic situations. To our knowledge, this is the first time that outdoor visual navigation with obstacle avoidance is carried out at approximately 1 ms^{-1} on over 500 m, using neither GPS nor maps. Perspective work includes using a more sophisticated obstacle model (e.g., with shape, dimension and velocity), to design an optimal/complete rather than a worst case algorithm.

ACKNOWLEDGMENT

This work has been supported by ANR (French National Agency) CityVIP project under grant ANR-07 TSFA-013-01.

REFERENCES

[1] U. Nunes, C. Laugier and M. Trivedi, "Introducing perception, planning, and navigation for Intelligent Vehicles" in *IEEE Trans. on Intelligent Transportation Systems*, vol. 10, no. 3, 2009, pp. 375–379.

[2] M. Buehler, K. Lagnemma and S. Singh (Editors), "Special Issue on the 2007 DARPA Urban Challenge, Part I-III", in *Journal of Field Robotics*, vol. 25, no. 8–10, 2008, pp. 423–860.

[3] T. Wada, S. Doi and S. Hiraoka, "A deceleration control method of automobile for collision avoidance based on driver's perceptual risk", *IROS*, 2009.

[4] T. Kato, Y. Ninomiya and I. Masaki, "An obstacle detection method by fusion of radar and motion stereo", in *IEEE Trans. on Intelligent Transportation Systems*, vol. 3, no. 3, 2002, pp. 182–188.

[5] A. Ohya, A. Kosaka and A. Kak, "Vision-based navigation by a mobile robot with obstacle avoidance using a single-camera vision and ultrasonic sensing", *IEEE Trans. on Robotics and Automation*, vol. 14, no. 6, 1998, pp. 969–978.

[6] Z. Yan, X. Xiaodong, P. Xuejun and W. Wei, "Mobile robot indoor navigation using laser range finder and monocular vision", *IEEE Int. Conf. on Robotics, Intelligent Systems and Signal Processing*, 2003.

[7] F. Lamiraux, D. Bonnafous and O. Lefebvre, "Reactive path deformation for nonholonomic mobile robots", in *IEEE Trans. on Robotics*, vol. 20, no. 6, 2004, pp. 967–977.

[8] T.-S. Lee, G.-H. Eoh, J. Kim and B.-H. Lee, "Mobile robot navigation with reactive free space estimation", *IROS*, 2010.

[9] A. Cherubini and F. Chaumette, "Visual navigation with obstacle avoidance", *IROS*, 2011.

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

[11] 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.

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

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

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

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

[16] 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*, vol. 25, no. 9, 2008, pp. 640–673.

[17] E. Royer, M. Lhuillier, M. Dhome and J.-M. Lavest, "Monocular vision for mobile robot localization and autonomous navigation", *Int. Journal of Computer Vision*, vol. 74, no. 3, 2007, pp. 237–260.