

Visual navigation of a mobile robot with laser-based collision avoidance

Andrea Cherubini, François Chaumette

▶ **To cite this version:**

Andrea Cherubini, François Chaumette. Visual navigation of a mobile robot with laser-based collision avoidance. The International Journal of Robotics Research, 2013, 32, pp.189-205. 10.1177/0278364912460413 . hal-00750623

HAL Id: hal-00750623

https://inria.hal.science/hal-00750623

Submitted on 11 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.

Visual Navigation of a Mobile Robot with Laser-based Collision Avoidance

Andrea Cherubini*[1,2] and François Chaumette†[1]

[1]INRIA Rennes - Bretagne Atlantique, IRISA, Campus de Beaulieu 35042, Rennes, France
[2]LIRMM - Université de Montpellier 2 CNRS, 161 Rue Ada, 34392 Montpellier, France

August 19, 2012

Abstract

In this paper, we propose and validate a framework for visual navigation with collision avoidance for a wheeled mobile robot. Visual navigation consists of following a path, represented as an ordered set of key images, which have been acquired by an on-board camera in a teaching phase. While following such path, the robot is able to avoid obstacles which were not present during teaching, and which are sensed by an on-board range scanner. Our control scheme guarantees that obstacle avoidance and navigation are achieved simultaneously. In fact, in the presence of obstacles, the camera pan angle is actuated to maintain scene visibility while the robot circumnavigates the obstacle. The risk of collision and the eventual avoiding behaviour are determined using a tentacle-based approach. The framework can also deal with unavoidable obstacles, which make the robot decelerate and eventually stop. Simulated and real experiments show that with our method, the vehicle can navigate along a visual path while avoiding collisions.

Keywords: Vision-based Navigation, Visual Servoing, Collision Avoidance, Integration of vision with other sensors.

*Andrea.Cherubini@lirmm.fr
†Francois.Chaumette@inria.fr

1 Introduction

A great amount of robotics research focuses on vehicle guidance, with the ultimate goal of automatically reproducing the tasks usually performed by human drivers [Buehler et al. 2008, Zhang et al. 2008, Nunes et al. 2009, Broggi et al. 2010]. In many recent works, information from visual sensors is used for localization [Guerrero et al. 2008, Scaramuzza and Siegwart 2008] or for navigation [Bonin-Font et al. 2008, López-Nicolás et al. 2010]. In the case of autonomous navigation, an important task is obstacle avoidance, which consists of either generating a collision-free trajectory to the goal [Minguez et al. 2008], or decelerating to prevent collision when bypassing is impossible [Wada et al. 2009]. Most obstacle avoidance techniques, particularly those that use motion planning [Latombe 1991], rely on the knowledge of a global and accurate map of the environment and obstacles.

Instead of utilizing such a global model of the environment, which would infringe the *perception to action* paradigm [Sciavicco and Siciliano 2000], we propose a framework for obstacle avoidance with simultaneous execution of a *visual servoing* task [Chaumette and Hutchinson 2006]. Visual servoing is a well known method that uses vision data directly in the control loop, and that has been applied on mobile robots in many works [Mariottini et al. 2007, Becerra et al. 2011, López-Nicolás and Sagüés 2011, Allibert et al. 2008]. For example, in [Mariottini et al. 2007] the epipolar geometry is exploited to drive a nonholonomic robot to a desired configuration. A similar approach is presented in [Becerra et al. 2011], where the singularities are dealt with more efficiently. The same authors achieve vision-based pose stabilization using a state observer in [López-Nicolás and Sagüés 2011]. Trajectory tracking is tackled in [Allibert et al. 2008] by integrating differential flatness and predictive control.

The visual task that we focus on is appearance-based navigation, which has been the target of our research in [Šegvić et al. 2008, Cherubini et al. 2009, Diosi et al. 2011]. In the framework that we have developed in the past[1], the path is a topological graph, represented by a database of ordered key images. In contrast with other similar approaches, such as [Booij et al. 2007], our graph does not contain forks. Furthermore, as opposed to [Royer et al. 2007, Goedemé et al. 2007, Zhang and Kleeman 2009, Fontanelli et al. 2009, Courbon et al. 2009], we do not use the robot pose for navigating along the path. Instead, our task is purely image-based (as in [Becerra et al. 2010]), and it is divided into a series of subtasks, each consisting of driving the robot towards the next key image in the database. More importantly, to our knowledge, appearance-based navigation frameworks have never been extended to take into account obstacles.

[1] See: www.irisa.fr/lagadic/demo/demo-cycab-vis-navigation/vis-navigation.

Obstacle avoidance has been integrated in many model-based navigation schemes. In [Yan et al. 2003], a range finder and monocular vision enable navigation in an office environment. The desired trajectory is deformed to avoid sensed obstacles in [Lamiraux et al. 2004]. The authors of [Ohya et al. 2008] use a model-based vision system with retroactive position correction. Simultaneous obstacle avoidance and path following are presented in [Lapierre et al. 2007], where the geometry of the path (a curve on the ground) is perfectly known. In [Lee et al. 2010], obstacles are circumnavigated while following a path; the radius of the obstacles (assumed cylindrical) is known a priori. In practice, all these methods are based on the environment 3D model, including, for example, walls and doors, or on the path geometry. In contrast, we propose a navigation scheme which does not require neither the environment nor the obstacle model.

One of the most common techniques for model-free obstacle avoidance is the potential field method, originally introduced in [Khatib 1985]. The gap between global path planning and real-time sensor-based control has been closed with the elastic band [Quinlan and Khatib 1993], a deformable collision-free path, whose initial shape is generated by a planner, and then deformed in real time, according to the sensed data. Similarly, in [Bonnafous et al. 2001, Von Hundelshausen et al. 2008], a set of trajectories (arcs of circles or "tentacles") is evaluated for navigating. However, in [Bonnafous et al. 2001], a sophisticated probabilistic elevation map is used, and the selection of the optimal tentacle is based on its risk and interest, which both require accurate pose estimation. Similarly, in [Von Hundelshausen et al. 2008], the trajectory computation relies on GPS way points, hence - once more - on the robot pose.

Here, we focus on this problem: a wheeled vehicle, equipped with an actuated pinhole camera and with a forward-looking range scanner, must follow a visual 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 [Kato et al. 2002], only one sensor is used to detect the obstacles). In this sense, our work is similar to [Folio and Cadenat 2006], where redundancy enables reactive obstacle avoidance, without requiring any 3D model. A robot is redundant when it has more DOFs than those required for the primary task; then, a secondary task can also be executed. In [Folio and Cadenat 2006], the two tasks are respectively visual servoing and obstacle avoidance. However, there are various differences with that work. First, we show that the redundancy approach is not necessary, since we design the two tasks so that they are independent. Second, we can guarantee asymptotic stability of the visual task at all times, in the presence of non occluding obstacles. Moreover, our controller is compact, and the transitions between safe and unsafe contexts is operated only for obstacle avoidance, while in [Folio and Cadenat 2006], three

controllers are needed, and the transitions are more complex. This compactness leads to smoothness of the robot behaviour. Finally, in [Folio and Cadenat 2006], a positioning task in indoor environments is considered, whereas we aim at continuous navigation on long outdoor paths.

Let us summarize the other major contributions of our work. An important contribution is that our approach is merely appearance-based, hence simple and flexible: the only information required is the database of key images, and no model of the environment or obstacles is necessary. Hence, there is no need for sensor data fusion nor planning, which can be computationally costly, and requires precise calibration of the camera/scanner pair. We guarantee that the robot will never collide in the case of static, detectable obstacles (in the worse cases, it will simply stop). We also prove that our control law is always well-defined, and that it does not present any local minima. To our knowledge, this is the first time that obstacle avoidance and visual navigation merged directly at the control level (without the need for sophisticated planning) are validated in real outdoor urban experiments.

The framework that we present here is inspired from the one designed and validated in our previous work [Cherubini and Chaumette 2011]. However, many modifications have been applied, in order to adapt that controller to the real world. First, for obstacle avoidance, we have replaced classical potential fields with a new tentacle-based technique inspired from [Bonnafous et al. 2001] and [Von Hundelshausen et al. 2008], which is perfectly suitable for appearance-based tasks, such as visual navigation. In contrast with those works, our approach does not require the robot pose, and exploits the robot geometric and kinematic characteristics (this aspect will be detailed later in the paper). A detailed comparison between the potential field and the tentacle techniques is given in [Cherubini et al. 2012]. In that work, we showed that with tentacles, smoother control inputs are generated, higher velocities can be applied, and only dangerous obstacles are taken into account. In summary, the new approach is more robust and efficient than its predecessor. A second modification with respect to [Cherubini and Chaumette 2011] concerns the design of the translational velocity, which has been changed to improve visual tracking and avoid undesired deceleration in the presence of non-dangerous obstacles. Another important contribution of the present work is that, in contrast with the tentacle-based approaches designed in [Von Hundelshausen et al. 2008] and [Bonnafous et al. 2001], our method does not require the robot pose. Finally, the present article reports experiments, which, for the first time in the field of visual navigation with obstacle avoidance, have been carried out in real-life, unpredictable urban environments.

The article is organized as follows. In Section 2, the characteristics of our problem (visual path following

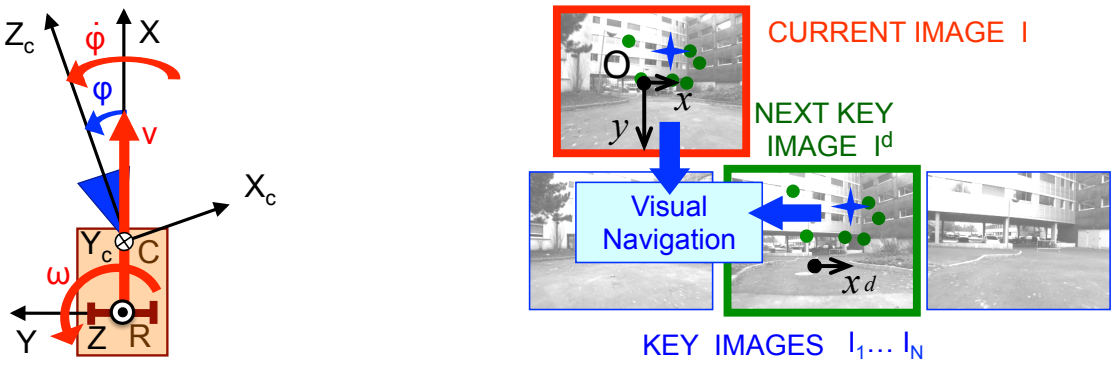


Figure 1: General definitions. Left: top view of the robot (rectangle), equipped with an actuated camera (triangle); the robot and camera frame (respectively, $\mathcal{F}_{\mathcal{R}}$ and $\mathcal{F}_{\mathcal{C}}$) are shown. Right: database of the key images, with current and next key images emphasized; the image frame $\mathcal{F}_{\mathcal{I}}$ is also shown, as well as the visual features (circles) and their centroid (cross).

with simultaneous obstacle avoidance) are presented. The control law is presented in full details in Section 3, and a short discussion is carried out in Section 4. Simulated and real experimental results are presented respectively in Sections 5 and 6, and summarized in the conclusion.

2 Problem Definition

2.1 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), image frame $\mathcal{F}_{\mathcal{I}}(O, x, y)$ (O is the image center), and camera frame $\mathcal{F}_{\mathcal{C}}\,(C, X_c, Y_c, Z_c)$ (C is the optical center). The robot control inputs are:

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

These are, respectively, the translational and angular velocities of the vehicle, and the camera pan angular velocity. We use the normalized perspective camera model:

$$x = \frac{X_c}{Z_c}, \qquad\qquad y = \frac{Y_c}{Z_c}.$$

We assume that the camera pan angle is bounded: $|\varphi| \leq \frac{\pi}{2}$, and that C belongs to the camera pan rotation axis, and to the robot sagittal plane (i.e., the plane orthogonal to the ground through X). We also assume that the path can be followed with continuous $v\,(t) > 0$. This ensures safety, since only obstacles in front of the robot can be detected by our range scanner.

2.2 Visual Path Following

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, right) is selected. Then, during autonomous navigation, the current image, noted I, is compared with the next key image in the database, $I^d \in \{I_1, \ldots, I_N\}$, 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, as well as visual point detection and tracking, we use the algorithm presented in [Royer et al. 2007]. 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 towards 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.

2.3 Obstacle Representation

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. 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 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. The size of the grid should increase with the robot velocity, to guarantee the sufficient time for obstacle avoidance. An appropriate choice for $|X_m|$ is the length of the robot, since obstacles behind cannot hit it as it advances. In this work, we use: $X_M = Y_M = 10$ m, $X_m = -2$ m, $Y_m = -10$ m. Any grid cell c centered at (X, Y) is considered occupied if an obstacle has been sensed in c. The cells have size 0.2×0.2 m. For the cells entirely lying in the scanner area, only the current scanner reading is considered. For all other cells in the grid, we use past readings, which are progressively displaced using odometry.

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

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

a set of drivable paths (tentacles). Each tentacle j is a semicircle that starts in R, is tangent to X, and is

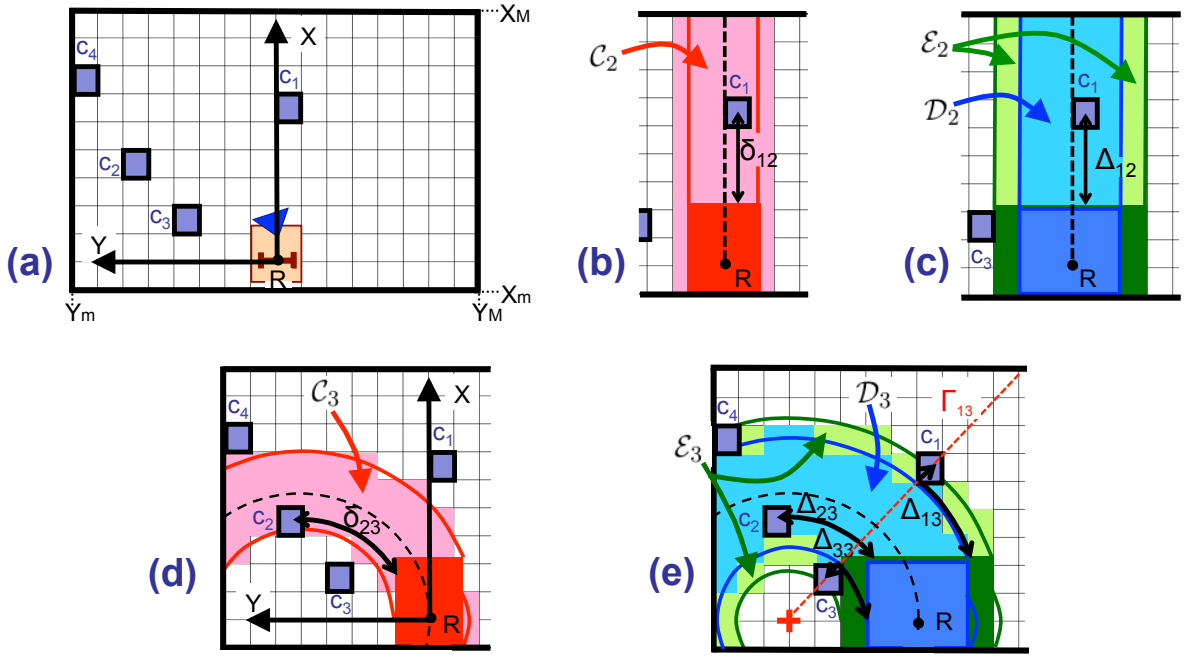


Figure 2: Obstacle models with 4 occupied cells $c_1, \ldots, c_4$. (a) Occupancy grid, straight (b, c) and sharpest counterclockwise (d, e) tentacles (dashed). When a total of 3 tentacles is used, the straight and sharpest counterclockwise are characterized respectively by index $j = 2$ and $j = 3$. For these two tentacles, we have drawn: classification areas (collision $\mathcal{C}_j$, dangerous central $\mathcal{D}_j$, dangerous external $\mathcal{E}_j$), corresponding boxes and delimiting arcs of circle, and cell risk and collision distances (Δ_{ij}, δ_{ij}). For tentacle $j = 3$ in the bottom right, we have also drawn the tentacle center (cross) and the ray of cell c_1, denoted Γ_{13}.

characterized by its curvature (i.e., inverse radius) κ_j, which belongs to $\mathcal{K}$, a uniformly sampled set:

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

The maximum desired curvature $\kappa_M > 0$, must be feasible considering the robot kinematics. Since, as we will show, 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 $\mathcal{K}$, i.e., its sampling interval. Indeed, a large set is costly since, as we show later, various collision variables must be calculated on each tentacle. On the other hand, extending the set enhances motion precision, since more alternative tentacles can be selected for navigation. In the simulations and experiments, we used 21 tentacles. In Fig. 2(b-e), the straight and the sharpest counterclockwise ($\kappa = \kappa_M$) tentacle are dashed. 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*), which are obtained by rigidly displacing, along the tentacle, three rectangular boxes, with

increasing size. The boxes are all overestimated with respect to the real robot dimensions. For each tentacle j, the sets of cells belonging to the three classification areas (shown in Fig. 2) are noted $\mathcal{C}_j$, $\mathcal{D}_j$ and $\mathcal{E}_j$. Cells belonging to the *dangerous central* set, are not considered in the *dangerous external* set as well, so that $\mathcal{D}_j \bigcap \mathcal{E}_j = \emptyset$. The sets $\mathcal{O}$, $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{E}$ are used to calculate the variables required in the control law defined in Section 3.1: in particular, the largest classification areas $\mathcal{D}$ and $\mathcal{E}$ are used to select the safest tentacle and its risk, while the thinnest one $\mathcal{C}$ determines the eventual necessary deceleration.

In summary, as we mentioned in Sect 1, our tentacles exploit the robot geometric and kinematic characteristics. Specifically, the robot geometry (i.e., the vehicle encumbrance) defines the three classification areas $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{E}$, hence the cell potential danger, while the robot kinematics (i.e., the maximum desired curvature, κ_M) define the bounds on the set of tentacles $\mathcal{K}$. Both aspects give useful information on possible collisions with obstacles ahead of the robot, which will be exploited, as we will show in Sect. 3, to choose the best tentacle and to eventually slow down or stop the robot.

2.4 Task Specifications

Let us recall the Jacobian paradigm which relates a robot kinematic control inputs with the desired task. We name $\mathbf{s} \in I\!R^m$ the task vector, and $\mathbf{u} \in I\!R^m$ the control inputs. The task dynamics are related to the control inputs by:

$$\dot{\mathbf{s}} = \mathbf{J}\mathbf{u}, \tag{1}$$

where $\mathbf{J}$ is the *task jacobian* of size $m \times m$. In this work, $m = 3$, and the desired specifications are:

1. orienting the camera in order to drive the abscissa of the feature centroid x to its value at the next key image in the database x^d,
2. making the vehicle progress forward along the path (except if obstacles are unavoidable),
3. avoiding collision with the obstacles, while remaining near the 3D taught path.

The required task evolution can be written:

$$\dot{\mathbf{s}}^* = \dot{\mathbf{s}}^d - \Lambda \left(\mathbf{s} - \mathbf{s}^d \right), \tag{2}$$

with $\mathbf{s}^d$ and $\dot{\mathbf{s}}^d$ indicating the desired values of the task, and of its first derivative, and $\Lambda = \text{diag}\ (\lambda_1 \dots \lambda_m)$ a positive definite diagonal gain matrix.

Since we assume that the visual features are static, the first specification on camera orientation can be expressed by:

$$\dot{x}^* = -\lambda_x \left(x - x^d \right), \tag{3}$$

with λ_x a positive scalar gain. This guarantees that the abscissa of the centroid of the points converges exponentially to its value at the next key image x^d, with null velocity there ($\dot{x}^d = 0$). The dynamics of this task can be related to the robot control inputs by:

$$\dot{x} = \mathbf{J}_x \mathbf{u} = \left[\begin{array}{ccc} j_v & j_\omega & j_{\dot{\varphi}} \end{array} \right] \mathbf{u}, \tag{4}$$

where j_v, j_ω and $j_{\dot{\varphi}}$ are the components of the *centroid abscissa Jacobian* $\mathbf{J}_x$ related to each of the three robot control inputs. Their form will be determined in Section 3.2.

The two other specifications (vehicle progression with collision avoidance) are related to the danger represented by the obstacles present in the environment. If it is possible, the obstacles should be circumnavigated. Otherwise, the vehicle should stop to avoid collision. To determine the best behaviour, we assess the danger at time t with a *situation risk function* $H : \mathbb{R}^{*+} \mapsto [0, 1]$, that will be fully defined in Section 3.3.

- In the *safe context* ($H = 0$), no dangerous obstacles are detected on the robot path. In this case, it is desirable that the robot acts as in the teaching phase, i.e., following the taught path with the camera looking forward. If the current pan angle φ is non-null, which is typically the case when the robot has just avoided an obstacle, an exponential decrease of φ is specified. Moreover, the translational velocity v must be reduced in the presence of sharp turns, to ease the visual tracking of quickly moving features in the image; we specify this using a function v_s that will be detailed in Section 3.4. In summary, the specifications in the safe context are:

$$\left\{ \begin{array}{l} \dot{x} = -\lambda_x \left(x - x^d \right) \\ v = v_s \\ \dot{\varphi} = -\lambda_\varphi \varphi \end{array} \right. , \tag{5}$$

with λ_φ a positive scalar gain. The corresponding current and desired task dynamics are:

$$\dot{\mathbf{s}}_s = \left[\begin{array}{c} \dot{x} \\ v \\ \dot{\varphi} \end{array} \right], \qquad \dot{\mathbf{s}}_s^* = \left[\begin{array}{c} -\lambda_x \left(x - x^d \right) \\ v_s \\ -\lambda_\varphi \varphi \end{array} \right]. \tag{6}$$

Using (4) we can derive the Jacobian relating $\dot{\mathbf{s}}_s$ and $\mathbf{u}$:

$$\dot{\mathbf{s}}_s = \mathbf{J}_s \mathbf{u} = \left[\begin{array}{ccc} j_v & j_\omega & j_{\dot{\varphi}} \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{array} \right] \mathbf{u}. \tag{7}$$

Note that matrix $\mathbf{J}_s$ is invertible if $j_\omega \neq 0$, and we will see in Section 3.2 that this condition is indeed ensured.

- In the *unsafe context* ($H = 1$), dangerous obstacles are detected. The robot should circumnavigate them by following the *best tentacle* (selected considering both the visual and avoidance tasks as we will see in Section 3.3). This heading variation drives the robot away from the 3D taught path. Correspondingly, the camera pan angle must be actuated to maintain visibility of the database features, i.e., to guarantee (3). The translational velocity must be reduced for safety reasons (i.e., to avoid collisions); we specify this using a function v_u, that will be defined in Section 3.5. In summary, the specifications in the unsafe context are:

$$\begin{cases} \dot{x} = -\lambda_x \left(x - x^d\right) \\ v = v_u \\ \omega = \kappa_b v_u \end{cases}, \tag{8}$$

where κ_b is the best tentacle curvature, so that the translational and angular velocities guarantee that the robot precisely follows it, since: $\omega/v = \kappa_b$. The current and desired task dynamics corresponding to (8) are:

$$\dot{\mathbf{s}}_u = \begin{bmatrix} \dot{x} \\ v \\ \omega \end{bmatrix}, \qquad \dot{\mathbf{s}}_u^* = \begin{bmatrix} -\lambda_x \left(x - x^d\right) \\ v_u \\ \kappa_b v_u \end{bmatrix}. \tag{9}$$

Using (4) we can derive the Jacobian relating $\dot{\mathbf{s}}_u$ and $\mathbf{u}$:

$$\dot{\mathbf{s}}_u = \mathbf{J}_u \mathbf{u} = \begin{bmatrix} j_v & j_\omega & j_{\dot{\varphi}} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{u}. \tag{10}$$

Matrix $\mathbf{J}_u$ is invertible if $j_{\dot{\varphi}} \neq 0$, and we will see in Section 3.2 that this condition is also ensured.

- In *intermediate contexts* ($0 < H < 1$), the robot should navigate between the taught path, and the best tentacle. The transition between these two extremes will be driven by situation risk function H.

3 Control Scheme

3.1 General Scheme

An intuitive choice for controlling (1) in order to fulfill the desired tasks $\mathbf{s}_u$ and $\mathbf{s}_s$ would be:

$$\mathbf{u} = \mathbf{J}^{-1} \dot{\mathbf{s}}^* \tag{11}$$

with:

$$\mathbf{s} = H\mathbf{s}_u + (1-H)\,\mathbf{s}_s$$

and therefore (considering $\dot{H} = 0$):

$$\mathbf{J} = H\mathbf{J}_u + (1-H)\,\mathbf{J}_s = \begin{bmatrix} j_v & j_\omega & j_{\dot{\varphi}} \\ 1 & 0 & 0 \\ 0 & H & 1-H \end{bmatrix}$$

In fact, away from singularities of $\mathbf{J}$, controller (11) leads to the linear system:

$$\dot{\mathbf{s}} - \dot{\mathbf{s}}^d = -\Lambda\left(\mathbf{s} - \mathbf{s}^d\right)$$

for which, as desired, $\left(\mathbf{s}^d, \dot{\mathbf{s}}^d\right)$ are exponentially stable equilibria, for any value of $H \in [0,\ 1]$ (since Λ is a positive definite diagonal matrix). Note that replacing (2) in (11), leads to the well known controller for following trajectory $s^d = s^d(t)$, given in [Chaumette and Hutchinson 2007]:

$$u = -\Lambda J^{-1}(s - s^d) + J^{-1}\dot{s}^d.$$

However, the choice of controller (11), is not appropriate for our application, since $\mathbf{J}$ is singular whenever:

$$Hj_{\dot{\varphi}} + (H-1)\,j_\omega = 0 \tag{12}$$

This condition, which depends on visual variables ($j_{\dot{\varphi}}$ and j_ω) as well as on an obstacle variable (H), can occur in practice.

Instead, we propose the following control law to guarantee a smooth transition between the inputs:

$$\mathbf{u} = H\mathbf{J}_u^{-1}\dot{\mathbf{s}}_u^* + (1-H)\,\mathbf{J}_s^{-1}\dot{\mathbf{s}}_s^* \tag{13}$$

Replacing this equation in (7) and (10), guarantees that controller (13) leads to convergence to the desired tasks (5) and (8):

$$\begin{array}{ll} \dot{\mathbf{s}}_s = \dot{\mathbf{s}}_s^* & \text{if } H = 0 \\ \dot{\mathbf{s}}_u = \dot{\mathbf{s}}_u^* & \text{if } H = 1 \end{array}$$

and that, in these cases, the desired states are globally asymptotically stable for the closed loop system. In Section 4, we will show that global asymptotic stability of the visual task is also guaranteed in the intermediate cases ($0 < H < 1$).

In the following, we will define the variables introduced in Section 2.4. We will show how to derive the centroid abscissa Jacobian $\mathbf{J}_x$, the situation risk function H, the best tentacle along with its curvature κ_b, and the translational velocities in the safe and unsafe context (respectively v_s and v_u).

3.2 Jacobian of the Centroid Abscissa

We will hereby derive the components of $\mathbf{J}_x$ introduced in (4). Let us define: $\mathbf{v} = (v_c,\ \omega_c)$ the camera velocity, expressed in $\mathcal{F}_\mathcal{C}$. Since we have assumed that the features are static, the dynamics of x can be related to $\mathbf{v}$ by:

$$\dot{x} = \mathbf{L}_x \mathbf{v}$$

where $\mathbf{L}_x$ is the interaction matrix of x [Chaumette and Hutchinson 2006]. In the case of a point of depth Z_c, it is given by [Chaumette and Hutchinson 2006]:

$$\mathbf{L}_x = \left[\begin{array}{cccccc} -\frac{1}{Z_c} & 0 & \frac{x}{Z_c} & xy & -1-x^2 & y \end{array} \right] \tag{14}$$

In theory, since we consider the centroid and not a physical point, we should not use (14) for the interaction matrix, but the exact and more complex form given in [Tahri and Chaumette 2005]. However, using (14) provides a sufficiently accurate approximation [Cherubini et al. 2009]. It also has the strong advantage that it is not necessary to estimate the depth of all points, using techniques such as those described in [Davison et al. 2007, De Luca et al. 2008, Durand et al. 2010]. Only an approximation of Z_c, i.e., one scalar, is sufficient. In practice, we set a constant fixed value. This strategy has proved successful for visual navigation in [Cherubini et al. 2009].

For the robot model that we are considering, the camera velocity $\mathbf{v}$ can be expressed in function of $\mathbf{u}$ by using the geometric model:

$$\mathbf{v} = {}^C\mathbf{T}_R \mathbf{u}$$

with:

$${}^C\mathbf{T}_R = \left[\begin{array}{ccc} \sin\varphi & -X^C\cos\varphi & 0 \\ 0 & 0 & 0 \\ \cos\varphi & X^C\sin\varphi & 0 \\ 0 & 0 & 0 \\ 0 & -1 & -1 \\ 0 & 0 & 0 \end{array} \right]$$

In this matrix, X^C is the abscissa of the optical center C in the robot frame $\mathcal{F}_\mathcal{R}$. This parameter is specific of the robot platform. Since C belongs to the robot sagittal plane, and since the robot is constrained on the ground plane, this is the only coordinate of C in $\mathcal{F}_\mathcal{R}$ required for visual servoing.

Then, multiplying $\mathbf{L}_x$ by ${}^C\mathbf{T}_R$, we obtain the components of $\mathbf{J}_x$:

$$\begin{aligned} j_v &= \tfrac{-\sin\varphi + x\cos\varphi}{Z_c} \\ j_\omega &= \tfrac{X^C(\cos\varphi + x\sin\varphi)}{Z_c} + 1 + x^2 \\ j_{\dot{\varphi}} &= 1 + x^2. \end{aligned} \tag{15}$$

From (15) it is clear that $j_{\dot{\varphi}} \geq 1 \; \forall x \in I\!R$; hence, $\mathbf{J}_u$ is never singular (see (10)). Furthermore, it is possible to ensure that $j_\omega \neq 0$, so that $\mathbf{J}_s$ is also invertible (see (7)). In fact, in (15) we can guarantee that $j_\omega \neq 0$, by setting $Z_c > \frac{X^C}{2}$ in the $\mathbf{J}_x$ components. Indeed, condition $j_\omega \neq 0$ is equivalent to:

$$\frac{X^C \left(\cos\varphi + x \sin\varphi\right)}{Z_c} + 1 + x^2 \neq 0 \tag{16}$$

Since $|\varphi| \leq \frac{\pi}{2}$: $\cos\varphi + x\sin\varphi \geq -x, \forall x \in I\!R$. Hence, a sufficient condition for (16) is:

$$x^2 - \frac{X^C}{Z_c} x + 1 > 0$$

which occurs $\forall x \in I\!R$ when $\frac{X^C}{Z_c} < 2$. In practice, this condition can be guaranteed, since X^C is an invariant characteristic of the robot platform, and Z_c is a tunable control parameter, which can be set to a value greater than $\frac{X^C}{2}$. Besides, the value of X^C on most robots platforms is usually smaller than 1 m, which is much less than the scene depth in outdoor environments. In [Cherubini et al. 2009], we have shown that overestimating Z_c does not jeopardize navigation.

On the other hand, we can infer from (15) that the singularity of controller (11), expressed by (12) can occur frequently. For example, whenever Z_c is large, yielding $j_{\dot{\varphi}} \approx j_\omega$, and concurrently $H \approx 0.5$, $\mathbf{J}$ becomes singular. This confirms the great interest in choosing control scheme (13), which is always well-defined if $Z_c > \frac{X^C}{2}$.

3.3 Situation Risk Function and Best Tentacle

To derive the situation risk function H used in (13), we first calculate a candidate risk function $H_j \in [0, 1]$ for each tentacle, as will be explained below. Each 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 would cover along tentacle j before touching the cell center. For occupied cells in the external set, only a subset $\bar{\mathcal{E}}_j$ is taken into account: $\bar{\mathcal{E}}_j \subseteq \mathcal{O} \bigcap \mathcal{E}_j$. This subset contains only 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 $\bar{\mathcal{E}}_3 = \{c_1, c_3\}$. Cell c_4 is not considered dangerous, since it is external, and does not have a counterpart on the other side of the tentacle. 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 four occupied cells. Note that for a given cell, Δ_{ij} may have different values (or even be undefined) according to the tentacle that is considered.

When all risk distances on tentacle j 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 hand tuned thresholds Δ_d and Δ_s ($0 < \Delta_d < \Delta_s$), to design the tentacle risk function:

$$H_j = \begin{cases} 0 & \text{if } \Delta_j \geq \Delta_s \\ \frac{1}{2}\left[1 + \tanh\left(\frac{1}{\Delta_j - \Delta_d} + \frac{1}{\Delta_j - \Delta_s}\right)\right] & \text{if } \Delta_d < \Delta_j < \Delta_s \\ 1 & \text{if } \Delta_j \leq \Delta_d. \end{cases} \quad (17)$$

Note that H_j smoothly varies from 0, when the dangerous cells associated to tentacle j (if any) are far, to 1, when they are near. If $H_j = 0$, the tentacle is tagged as *clear*. In practice, threshold Δ_s must be set to the risk distance for which the context ceases to be safe (H becomes greater than 0), so that the robot starts to leave the taught – occupied – path. On the other hand, Δ_d must be tuned as the risk distance for which the context becomes unsafe ($H = 1$), so that the robot follows the best tentacle to circumnavigate the obstacle. In our work, we used the values $\Delta_s = 6$ m and $\Delta_d = 4.5$ m. The risk function H_j corresponding to these values is plotted in Fig. 3.

The H_js of all tentacles are then compared, in order to determine H in (13). 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 (13), 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 $v_s > 0$. We obviously constrain κ to the interval of feasible curvatures $[-\kappa_M, \kappa_M]$. Then, we derive the two neighbours 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[2]. We

[2]Without loss of generality, we consider that intervals are defined even when the first endpoint is greater than the second, e.g., $[\kappa_n, \kappa_{nn})$ should be read $(\kappa_{nn}, \kappa_n]$ if $\kappa_n > \kappa_{nn}$.

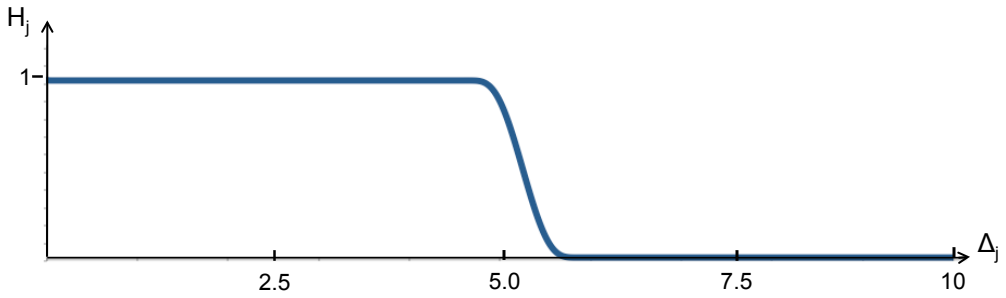


Figure 3: Risk H_j, in function of the tentacle risk distance Δ_j (m) when $\Delta_s = 6$ m and $\Delta_d = 4.5$ m.

denote it as the *visual task tentacle*. The situation risk function H_v of that tentacle is then obtained by linear interpolation of its neighbours:

$$H_v = \frac{(H_{nn} - H_n)\,\kappa + H_n\kappa_{nn} - H_{nn}\kappa_n}{\kappa_{nn} - \kappa_n}. \tag{18}$$

In practice, H_v measures the risk on the visual path, by considering only obstacles on the visual task tentacle and on its neighbour tentacle. In particular, for the context to be safe (i.e., in order to follow the taught path and realize the desired safe task in (6)), it is sufficient that the neighbour tentacles are clear ($H_n = H_{nn} = 0$). This way, obstacles on the sides do not deviate the robot away from the taught path.

Let us now detail our strategy for determining the best tentacle curvature κ_b for navigation. This strategy is illustrated by the four examples in Fig. 4, where 5 tentacles are used. In the figure, the dangerous cells (i.e., for each tentacle j, the cells in $\mathcal{D}_j \bigcup \mathcal{E}_j$) associated to the visual task tentacle and to the best tentacle are respectively shown in light and dark gray. The occupied cells are shown in black. The best tentacle is derived from the tentacle risk functions defined just above. If $H_v = 0$ (as in Fig. 4(a)), the visual task tentacle can be followed: we set $\kappa_b = \kappa_n$, and we apply (13) 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[3], noted κ_{pb}, and mid-gray in the figure. If many clear ones are present, the nearest to the visual task tentacle is chosen, as in Fig. 4(b). 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 (as in Fig. 4(c)), we select it and set $H = 0$. Instead, if no tentacle in $\mathcal{K}$ is clear, the one with minimum H_j calculated using (17) is chosen, and H is set equal to that H_j. In the example of Fig. 4(d), tentacle 1 is chosen and we set $H = H_1$, since $\Delta_1 = \sup\{\Delta_1, \ldots, \Delta_5\}$, hence $H_1 = \inf\{H_1, \ldots, H_5\}$. Eventual

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

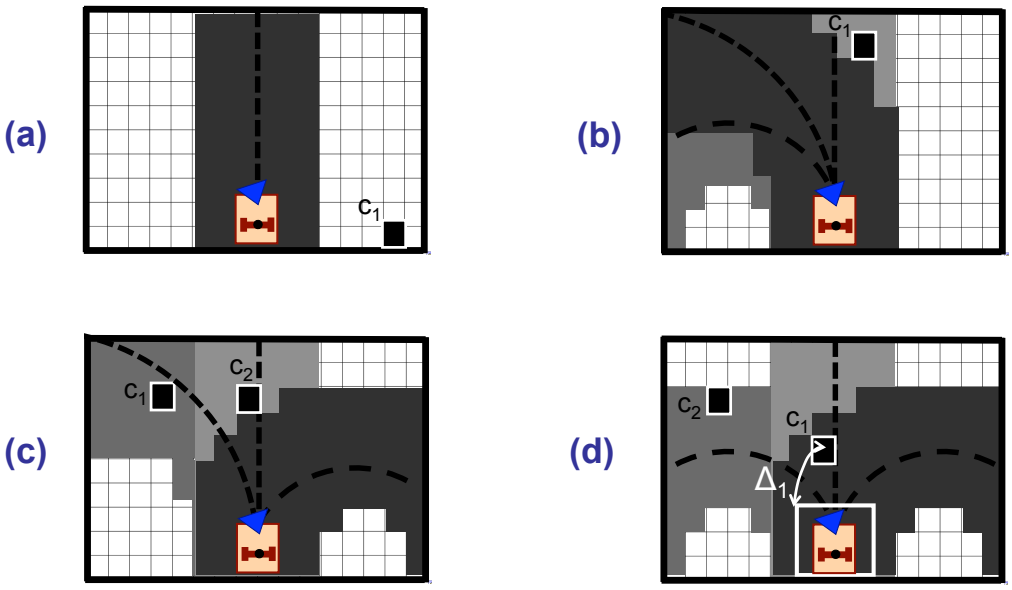


Figure 4: Strategy for selecting the best tentacle among 5 in four different scenarios. The cells associated to the visual task tentacle, to the previous best tentacle, and to the best tentacle are shown in increasingly dark gray; the corresponding tentacles are dashed, and the occupied cells c_1 and c_2 are shown in black. (a) Since it is clear, the visual task tentacle with curvature κ_n is selected: $\kappa_b = \kappa_n$. (b) The clear tentacle with curvature in $[\kappa_n, \kappa_{pb}]$ nearest to κ_n is chosen. (c) Since all tentacles with curvature in $[\kappa_n, \kappa_{pb}]$ are occupied, the clear one nearest to the visual task tentacle is chosen. (d) Since all tentacles are occupied, we select the one with smallest H_j, hence, largest risk distance Δ_j (here, Δ_1) given the middle boundary box).

ambiguities are again solved first with the distance from κ_n, then from κ_{nn}.

3.4 Translational Velocity in the Safe Context

We will hereby define the translational velocity in the safe context v_s. When the feature motion in the image is fast, the visual tracker is less effective, and the translational velocity should be reduced. This is typically the case at sharp robot turns, and when the camera pan angle φ is strong (since the robot is far from the taught 3D path). Hence, we define v_s as:

$$v_s(\omega, \varphi) = v_m + \frac{v_M - v_m}{4}\,\sigma \tag{19}$$

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 (19) has an upper bound $v_M > 0$ (for $\varphi = \omega = 0$), and smoothly decreases to the lower bound $v_m > 0$, as either $|\varphi|$ or $|\omega|$ grow. Both v_M and v_m are hand-tuned variables, and the decreasing trend is

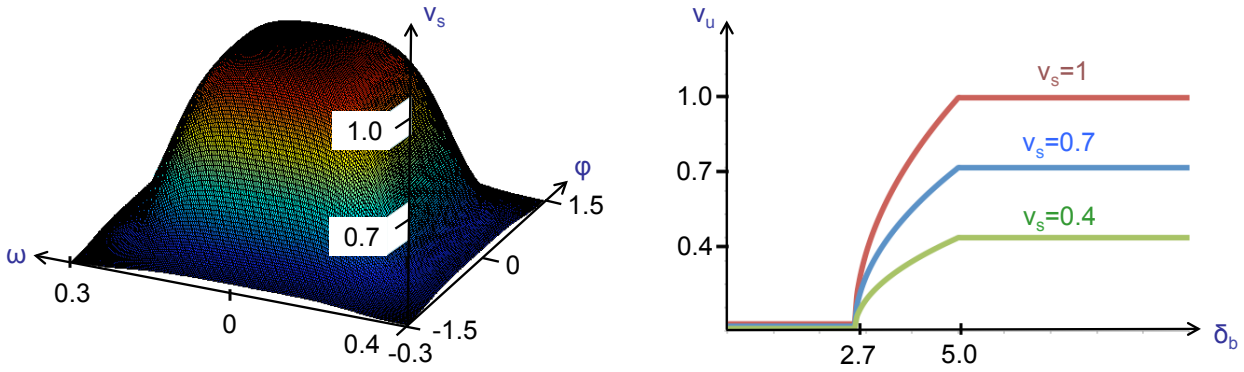


Figure 5: Left: safe translational velocity v_s (ms^{-1}) in function of ω (rads^{-1}) and φ (rad). Right: unsafe translational velocity v_u (ms^{-1}) in function of δ_b (m) for three different values of v_s.

determined by empirically tuned positive parameters k_ω and k_φ. This definition of v_s yields better results, both in terms of performances and smoothness than the one in [Cherubini and Chaumette 2011], which was only characterized by the image x variation. On the left of Fig. 5, we have plotted v_s for: $v_m = 0.4$ ms^{-1}, $v_M = 1$ ms^{-1}, $k_\omega = 13$, $k_\varphi = 3$.

3.5 Translational Velocity in the Unsafe Context

The unsafe translational velocity v_u must adapt to the potential danger; it is derived from the obstacles on the best tentacle, defined in Section 3.3. In fact, v_u is derived from the collision distance 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 box contains the robot, if R follows the best tentacle, collisions can only occur in occupied cells in $\mathcal{C}_b$. In fact, the collision with cell c_i will occur at the distance, denoted $\delta_{ib} \geq 0$, that the thinner 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 hand tuned thresholds such that $0 < \delta_d < \delta_s$. If the probable collision is far enough ($\delta_b \geq \delta_s$), the translational velocity can be maintained at the safe value defined in (19). Instead, if the dangerous occupied cell 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 the intermediate situations 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 general expression of the unsafe translational velocity becomes:

$$v_u(\delta_b) = \begin{cases} v_s & \text{if } \delta_b \geq \delta_s \\ v_s\sqrt{\delta_b - \delta_d/\delta_s - \delta_d} & \text{if } \delta_d < \delta_b < \delta_s \\ 0 & \text{if } \delta_b \leq \delta_d, \end{cases} \tag{20}$$

in order to decelerate as the collision distance δ_b decreases. In practice, threshold δ_d will be chosen as the distance to collision at which the robot should stop. Instead, threshold δ_s must be defined according to the maximum applicable deceleration (noted $a_M < 0$), in order to brake before reaching distance δ_d, even when the safe velocity v_s is at its maximum v_M:

$$\delta_s > \delta_d - \frac{2a_M}{v_M^2}.$$

In our work, we used $\delta_d = 2.7$ m and $\delta_s = 5$ m, as shown on the right of Fig. 5, where we have plotted v_u in function of δ_b for three values of v_s: 0.4, 0.7, and 1 ms^{-1}.

4 Discussion

In this section, we will instantiate and comment our control scheme for visual navigation with obstacle avoidance. Using all the variables defined above, we can explicitly write our controller (13) for visual navigation with obstacle avoidance:

$$\begin{cases} v = (1-H)\, v_s + H v_u \\ \omega = (1-H)\, \frac{\lambda_x\left(x^d - x\right) - j_v v_s + \lambda_\varphi j_{\dot\varphi} \varphi}{j_\omega} + H \kappa_b v_u \\ \dot\varphi = H \frac{\lambda_x\left(x^d - x\right) - (j_v + j_\omega \kappa_b) v_u}{j_{\dot\varphi}} - (1-H)\, \lambda_\varphi \varphi \end{cases}. \tag{21}$$

This control law has the following interesting properties.

1. In the *safe* context ($H = 0$), (21) becomes:

$$\begin{cases} v = v_s \\ \omega = \frac{\lambda_x\left(x^d - x\right) - j_v v_s + \lambda_\varphi j_{\dot\varphi} \varphi}{j_\omega} \\ \dot\varphi = -\lambda_\varphi \varphi \end{cases}. \tag{22}$$

In Sect. 3.1, we proved that this controller guarantees global asymptotic stability of the safe task $\dot{s}_s^*$. As in [Cherubini et al. 2009] and [Diosi et al. 2011], where obstacles were not considered, the image error is controlled only by ω, which also compensates the centroid displacements due to v and to $\dot{\varphi}$ through the image jacobian components (15), to fulfill the visual task (3). The two remaining specifications in (5), instead, are achieved by inputs v and $\dot{\varphi}$: the translational velocity is regulated to improve tracking according to (19), while the camera is driven forward, to $\varphi = 0$. Note that, to obtain $H = 0$ with the tentacle approach, it is sufficient that the neighbour tentacles are clear ($H_n = H_{nn} = 0$), whereas in the potential field approach used in [Cherubini and Chaumette 2011], 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.

2. In the *unsafe* context ($H = 1$), (21) becomes:

$$\begin{cases} v = v_u \\ \omega = \kappa_b v_u \\ \dot{\varphi} = \frac{\lambda_x \left(x^d - x\right) - (j_v + j_\omega \kappa_b) v_u}{j_{\dot{\varphi}}} \end{cases} . \tag{23}$$

In Sect. 3.1, we proved that this controller guarantees global asymptotic stability of the unsafe task $\dot{s}_u^*$. In this case, the visual task (3) is executed by $\dot{\varphi}$, while the two other specifications are ensured by the 2 other degrees of freedom: the translational velocity is reduced (and even zeroed to $v = v_u = 0$ for very near obstacles such that $\delta_b \leq \delta_d$), while the angular velocity makes the robot follow the best tentacle ($\omega / v = \kappa_b$). Note that, since no 3D positioning sensor (e.g., gps) is used, closing the loop on the best tentacle is not possible; however, even if the robot slips (e.g., due to a flat tire), at the following iterations tentacles with stronger curvature will be selected to drive it towards the desired path, and so on. Finally, the camera velocity $\dot{\varphi}$ in (23) compensates the robot rotation, to keep the features in view.

3. In *intermediate* situations ($0 < H < 1$), the robot navigates between the taught path, and the best path considering obstacles. The situation risk function H representing the danger on the taught path, drives the transition, but not the speed. In fact, note that, for all $H \in [0, 1]$, when $\delta_b \geq \delta_s$: $v = v_s$. Hence, a high velocity can be applied if the path is clear up to δ_s (e.g., when navigating behind another vehicle).

4. Control law (21) guarantees that obstacle avoidance has no effect on the visual task, which can be achieved for any $H \in [0, 1]$. Note that plugging the expressions of v, ω, and $\dot{\varphi}$ from (21) into the

visual task equation:

$$\dot{x} = j_v v + j_\omega \omega + j_{\dot{\varphi}} \dot{\varphi} \tag{24}$$

yields (3). Therefore, desired state x^d is globally asymptotically stable for the closed loop system, $\forall\ H \in [\,0,\ 1]$. This is true even in the special case where $v = 0$. In fact, the robot stops and v becomes null, if and only if $H = 1$ and $v_u = 0$, implying that $\omega = 0$ and $\dot{\varphi} = \frac{\lambda_x \left(x^d - x\right)}{j_{\dot{\varphi}}}$, which allows realization of the visual task. In summary, from a theoretical control viewpoint (i.e., without considering image processing nor field of view or joint limits constraints), this proves that if at least one point in I^d is visible, the visual task of driving the centroid abscissa to x^d will be achieved, even in the presence of unavoidable obstacles. This strategy is very useful for recovery: since the camera stays focused on the visual features, as soon as the path returns free, the robot can follow it again.

5. Controller (21) does not present local minima, i.e., non-desired state configurations for which $\mathbf{u}$ is null. In fact, when $H < 1$, $\mathbf{u} = 0$ requires both v_u and v_s to be null, but this is impossible since from (19)), $v_s > v_m > 0$. Instead, when $H = 1$, it is clear from (23) that for $\mathbf{u}$ to be null it is suffcient that $x^d - x = 0$ and $v_u = 0$. This corresponds to null desired dynamics: $\dot{\mathbf{s}}_u^* = 0$ (see (9)). This task is satisfied, since plugging $\mathbf{u} = 0$ into (10), yields precisely $\dot{\mathbf{s}}_u = 0 = \dot{\mathbf{s}}_u^*$.

6. If we tune the depth Z_c to infinity in (15), $j_v = 0$, and $j_{\dot{\varphi}} = j_\omega = 1 + x^2$. Thus, control law (21) becomes:

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

Note that, for small image error ($x \approx x^d$), $\dot{\varphi} \approx -\omega$. In practice, the robot rotation is compensated by the camera pan rotation, which is an expected behavior.

5 Simulations

In this section and in the following, we will detail the simulated and real experiments that were used to validate our approach. Simulations are in the video shown in Extension #1.

For simulations, we made use of Webots[4], where we designed a car-like robot equipped with an actuated 320×240 pixels 70° field of view camera, and with a 110° scanner of range 15 m. Both sensors operate at 30 Hz. The visual features, represented by spheres, are distributed randomly in the environment, with

[4] www.cyberbotics.com

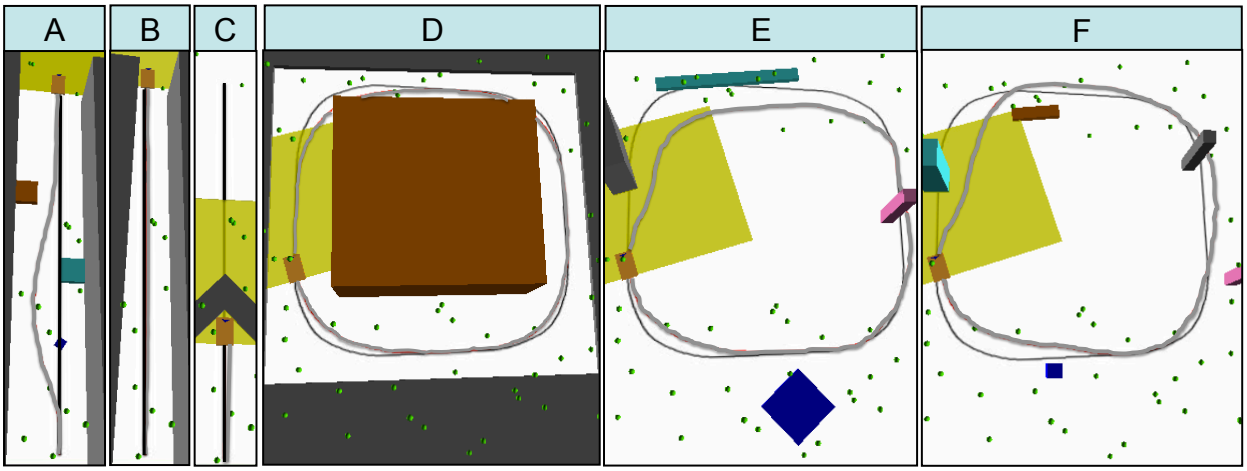


Figure 6: Six obstacle scenarios for replaying two taught paths (black) with the robot (rectangle): a straight segment (scenarios A to C), and a closed loop followed in the clockwise sense (D to F). Visual features are represented by the spheres, the occupancy grid by the rectangular area, and the replayed paths are drawn in gray.

depths with respect to the robot varying from 0.1 to 100 m. The offset between R and C is $X^C = 0.7$ m, and we set $Z_c = 15$ m that meets the condition $Z_c > \frac{X^C}{2}$. We use 21 tentacles, with $\kappa_M = 0.35$ m^{-1} (the robot maximum applicable curvature). For the situation risk function, we use $\Delta_s = 6$ m and $\Delta_d = 4.5$ m. These parameters correspond to the design of H shown in Fig. 3. The safe translational velocity is designed with $v_m = 0.4$ ms^{-1}, $v_M = 1$ ms^{-1}, $\kappa_\omega = 13$ and $\kappa_\varphi = 3$, as in the graph on the left of Fig. 5. For the unsafe translational velocity, we use $\delta_s = 5$ m, and $\delta_d = 2.7$ m as on the right of Fig. 5 (top curve). The simulations were helpful for tuning the control gains, in all experiments, to: $\lambda_x = 1$ and $\lambda_\varphi = 0.5$.

At first, no obstacle is present in the environment, and the robot is driven along a taught path. Then, up to 5 obstacles are located, near and on the taught path, and the robot must replay the visual path, while avoiding them. In addition, the obstacles may partially occlude the features. Although the sensors are noise-free, and feature matching is ideal, these simulations allow validation of controller (21).

By displacing the obstacles, we have designed the 6 scenarios shown in Fig. 6. For scenarios A, B and C, the robot has been previously driven along a 30 m straight path, and $N = 8$ key images have been acquired, whereas in scenarios D, E and F, the taught path is a closed loop of length 75 m and $N = 20$ key images, which is followed in the clockwise sense. In all scenarios, the robot is able to navigate without colliding, and this is done with the same parameters. The metrics used to assess the experiments are the image error with respect to the visual database $x - x^d$ (in pixels), averaged over the whole experiment and denoted $\bar{e}$,

and the distance, at the end of the experiment, from the final 3D key pose (ϵ, in cm). The first metric $\bar{e}$ is useful to assess the controller accuracy in realizing the visual path following task. The latter metric is less relevant, since task (3) is defined in the image space, and not in the pose space.

In all six scenarios, path following has been achieved, and in some cases, navigation was completed using only 3 image points. Obviously, this is possible in simulations, since feature detection is ideal: in the real case, which includes noise, 3 points may be insufficient. Some portions of the replayed paths, corresponding to the obstacle locations, are far from the taught ones. However, these deviations would have been indispensable to avoid collisions, even with a pose-based control scheme. Let us detail the robot behaviour in the six scenarios:

- Scenario A: two walls, which were not present during teaching, are parallel to the path, and three boxes are placed in between. The first box is detected, and overtaken on the left. Then, the vehicle passes between the second box and the left wall, and then overtakes the third box on the right. Finally, the robot converges back to the path, and completes it. Although the walls occlude features on the sides, the experiment is successful, with $\bar{e} = 5$, and $\epsilon = 23$.

- Scenario B: it is similar to A, except that there are no boxes, and that the left wall is shifted towards the right one, making the passage narrower towards the end. This makes the robot deviate in order to pass in the center of the passage. We obtain $\bar{e} = 6$, $\epsilon = 18$.

- Scenario C: this scenario is designed to test the controller in the presence of unavoidable obstacles. Two walls forming a corner, are located on the path. This soon makes all tentacles unsafe: $\Delta_j \leq \Delta_d| \quad \forall\, j$, yielding $H = 1$. Besides, as the robot approaches the wall, the collision distance on the best tentacle δ_b decreases, and eventually becomes smaller than δ_d, to make $v_u = 0$ and stop the robot (see (23)). Although the path is not completed (making metric ϵ irrelevant), the collision is avoided, and $\bar{e} = 4$ pixels. As proved in Section 4, convergence of the visual task ($x = x^d$) is achieved, in spite of $\mathbf{u} = 0$. In particular, here, the centroid abscissa on the third key image in the database is reached.

- Scenario D: high walls are present on both sides of the path; this leads to important occlusions (less than 50% of the database features are visible), and to a consequent drift from the taught path. Nevertheless, the final key image is reached, without collisions, and with $\bar{e} = 34$, and $\epsilon = 142$. Although this metric is higher than in the previous scenarios (since the path is longer and there are numerous

occlusions), it is still reasonably low.

- Scenario E: two obstacles are located on the path, and two other are near the path. The first obstacle is overtaken on the left, before avoiding the second one, also on the left. Then, the robot converges to the path and avoids the third obstacle on the right, before reaching the final key image. We obtain $\bar{e} = 33$, and $\epsilon = 74$. The experiment shows one of the advantages of our tentacle-based approach: lateral data in the grid is ignored (considering the fourth obstacle, would have made the robot curve away from the path).

- Scenario F: here, the controller is assessed in a situation where classical obstacle avoidance strategies (e.g. potential fields) often fail because of local minima. In fact, when detected, the first obstacle is centred on the X axis and orthogonal to it. This may induces an ambiguity, since occupancies are symmetric with respect to X. However, the visual features distribution, and consequent visual task tentacle κ_n drive the robot to the right of the obstacle, which is thus avoided. We have repeated this experiment with 10 randomly generated visual feature distributions, and in all cases the robot avoided the obstacle. The scenario involves four more obstacles, two of which are circumnavigated externally, and two on the inside. Here, $\bar{e} = 29$, and $\epsilon = 75$.

In Fig. 7, we show 5 stages of scenario A. In this figure, as well as later in Fig. 12, in Fig. 14, the segments linking the current and next key image points are drawn in the current image. In the occupancy grid, the dangerous cell sets associated to the visual task tentacle and to the best tentacle (when different) are respectively shown in gray and black, and two black segments indicate the scanner amplitude. Only cells that can activate H (i.e., cells at distance $\Delta < \Delta_s$) have been drawn. At the beginning of the experiment (iteration 10), the visual features are driving the robot towards the first obstacle. When it is near enough, the obstacle triggers H (iteration 200), forcing the robot away from the path towards the best tentacle, while the camera rotates clockwise to maintain feature visibility (iterations 200, 290 and 405). Finally (iteration 630), the controller drives the robot back to the path, and the camera to the forward-looking direction.

Further details can be obtained by studying some relevant variables. We focus on scenario E, for which we have plotted in Fig. 8 the values of H, v, ω, $\dot{\varphi}$, and φ during navigation. The curve of H shows when the obstacles, respectively the first (at iterations 0 – 1000), second (1100 – 1700) and third (3200 – 3700), have intervened in control law (21). As aforementioned, the fourth obstacle does not trigger H, since it is too far on the side to jeopardize the robot. Let us now discuss the trend of the five curves. Since the beginning, the

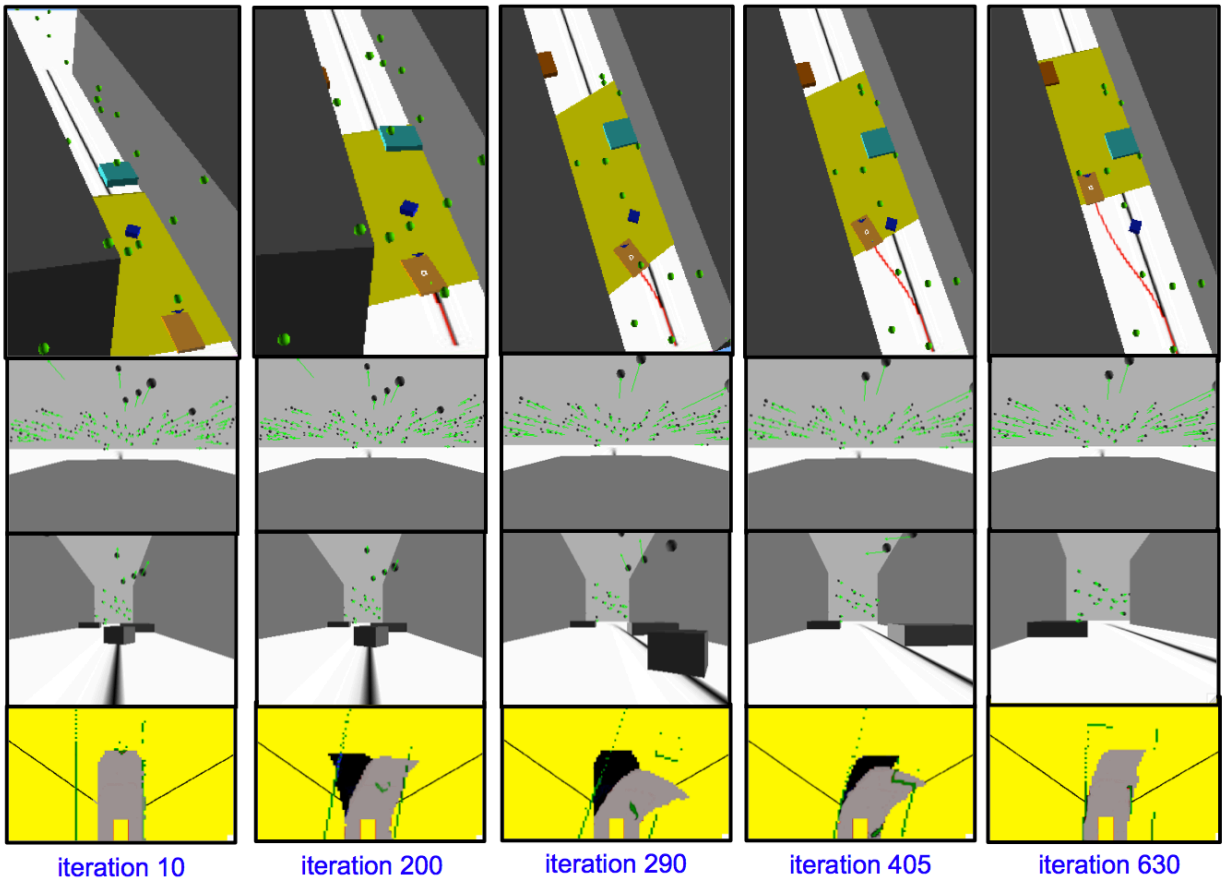


Figure 7: Scenario A. For each of the 5 relevant iterations we show (top to bottom): the robot overtaking the first obstacle, the next key image, the current image, and the occupancy grid.

first obstacle is detected: the tentacle selection induces a negative rotation on the robot (ω curve), a positive one on the camera ($\dot{\varphi}$), and a reduction of v. The strategy proves efficient, since the robot overtakes the obstacle. Soon afterwards, the second obstacle triggers H, and provokes a deceleration on v. Concurrently, the camera pan angle φ becomes positive to track the visual features which are mostly on the left of the robot (just like the taught path, as shown in Fig. 6). When the second obstacle is bypassed, the camera pan is reset to zero. The reduction of v at iteration 2800 is due only to the sharp turn (i.e., to the reduction of v_s), since the path is safe at this point. Then, the third obstacle triggers H, and is easily circumnavigated. From iteration 3700 onwards, the situation risk function is cancelled. Correspondingly, the variables are driven by (22). Note also that the camera angle φ is reset to 0 in less than 200 iterations, and remains null until the end of the experiment. The small spikes in the angular velocity ω, which appear throughout the experiment, correspond to the changes of database key images (except when they are provoked by the obstacles, as

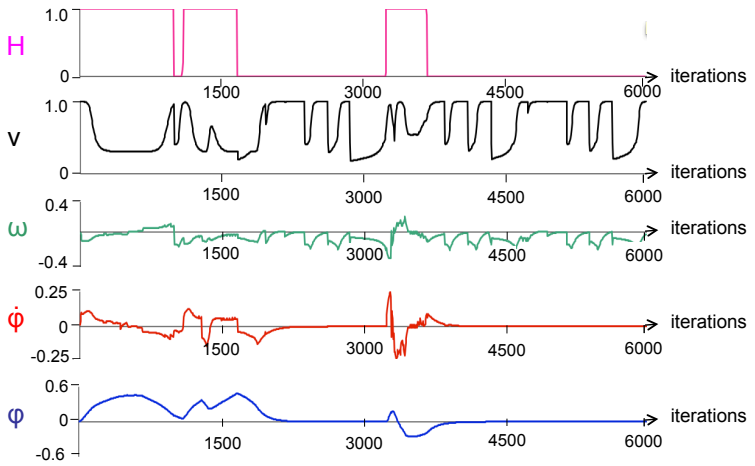


Figure 8: Evolution of relevant variables in scenario E; top to bottom: H, v (in m s^{-1}), ω (in rad s^{-1}), $\dot{\varphi}$ (in rad s^{-1}), and φ (in rad).

discussed above).

The six simulations have been repeated by setting the feature depth Z_c to infinity. For all six scenarios, the image accuracy, assessed with $\bar{e}$, is very near to the one obtained when $Z_c = 15$ m. On the other hand, the pose accuracy, assessed with ϵ, is lower when $Z_c = \infty$, as shown in Table 1. The difference is relevant on long paths (scenarios D, E and F). Although the navigation task is defined in the image space, these experiments show that tuning Z_c, even coarsely, according to the environment, can contribute to the controller performance in the 3D space. This aspect had already emerged in part in our previous work [Cherubini et al. 2009].

6 Real Experiments

After the simulations, the framework has been ported on our CyCab vehicle, set in car-like mode (i.e., using the front wheels for steering), for real outdoor experimental validation. The robot is equipped with a coarsely calibrated 320×240 pixels 70° field of view, B&W Marlin (F-131B) camera mounted on a TRACLabs Biclops Pan/Tilt head (the tilt angle is null, to keep the optical axis parallel to the ground), and with a 2-layer, 110° scanning angle, laser SICK LD-MRS. A dedicated encoder on the TRACLabs head precisely measures the pan angle φ required in our control law (see (21)). The grid is built by projecting the laser readings from the 2 layers on the ground. Exactly the same configuration (i.e., same parameters, gains and grid size) tuned in Webots is used on the real robot. The centroid depth value that we used in simulations ($Z_c = 15$ m) proved effective in all real experiments as well, although the scenarios were very

Table 1: Final 3D error ϵ (in cm) when $Z_c = 15$ m and $Z_c = \infty$ (for scenario C, since the path is not completed, ϵ is irrelevant).

scenario	A	B	D	E	F
$Z_c = 15$ m	23	18	142	74	75
$Z_c = \infty$	26	19	151	80	82

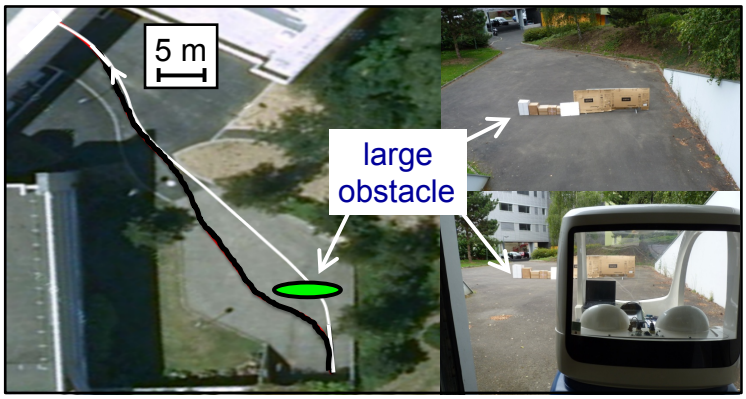


Figure 9: Scenario A (a long obstacle is avoided): taught (white) and replayed (black) paths.

variegate. This confirms, as shown in [Cherubini et al. 2009], that a very coarse approximation of the scene depth is sufficient to effectively tune Z_c. The velocity ($v_M = 1$, as in Webots) has been reduced due to the image processing rate (10 Hz), to limit the motion of features between successive images; the maximum speed attainable by the CyCab is 1.3 ms^{-1} anyway. Since camera (10 Hz) and laser (12.5 Hz) processing are not synchronized, they are implemented on two different threads, and the control input $\mathbf{u}$ derived from control law (21) is sent to the robot as soon as the visual information is available (10 Hz).

It is noteworthy to point out that the number of tentacles that must be processed, and correspondingly, the computational cost of the laser processing thread, increase with the context danger. For clarity, let us discuss two extreme cases: a safe and an occupied contexts. To verify that a context is safe (i.e., that $H_v = 0$ in (17)), all the cells in the dangerous areas $\mathcal{D} \bigcup \mathcal{E}$ of only the two neighbour tentacles must be explored. Instead, in a scenario where the grid is very occupied, all of the tentacles in $\mathcal{K}$ may need to be explored. In general, this second case will be more costly than the first. However, in practice, since only the minimum risk and collision distances (Δ_j and δ_b) are required by our controller, exploration of a tentacle stops as soon as the nearest cell is found occupied, so that the tentacles are rarely entirely explored. The experiments showed that the computational cost of laser processing, using the chosen number of tentacles (i.e., 21, as mentioned in Sect. 2.3), was never a critical issue with respect to that of image processing.

First, we have assessed the performance of our control scheme when a very long obstacle is positioned perpendicularly on the taught path (denoted path A, and shown in Fig. 9). In Fig. 10, we have plotted the control inputs $\mathbf{u}$, and the situation risk function H. The smooth trend of $\mathbf{u}$ at the beginning and end of the experiments is due to the acceleration saturation carried out at the CyCab low-level control. The obstacle is overtaken on the left, while the camera rotates right to maintain scene visibility (dotted black and dotted gray curves in Fig. 10). The robot is able to successfully reach the final key image and complete navigation, although it is driven over 5 meters away from the taught 3D path by the obstacle. In practice, soon after the obstacle is detected (i.e., after 5 s), tentacles with first positive ($5 - 16$ s), and then negative ($16 - 25$ s) curvature are selected. Since these tentacles are clear, v is reduced only for visual tracking, by (19) (solid black curve in Fig. 10). This is a major feature of the tentacle method, which considers only the actual collision risk of obstacles for reducing the translational velocity. 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 small counterclockwise rotation along with a slight decrease in v. Then the context returns safe, and the visual path can be followed for the rest of the experiment. The translational velocity averaged over the experiment is 0.79 ms^{-1}, which is more than twice the speed reached in [Cherubini and Chaumette 2011].

After these results, we have run a series of experiments, on longer and more crowded paths (denoted B to E on Fig. 11) on our campus. All campus experiments here are also visible in the video shown in Extension $\#2$. The Cycab was able to complete all paths (including 650 m long path E), while dealing with various natural and unpredictable obstacles, such as parked and driving cars and pedestrians. The translational velocity averaged over these experiment was 0.85 ms^{-1}.

Again, 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 aspect is clear from Fig. 12(left), where a stage of the navigation experiment on path E is illustrated. From top to bottom, we show: the next key image in the database I_d, the current image I, and three consecutive occupancy grids processed at that point of

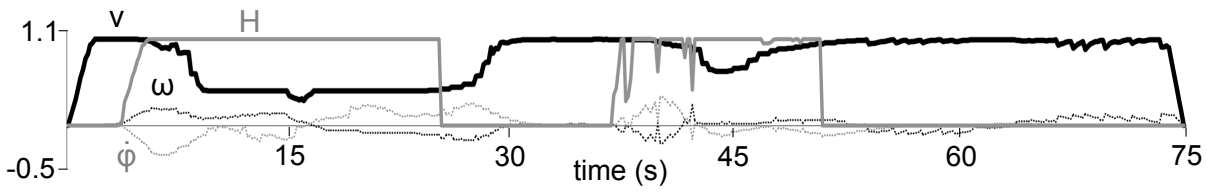


Figure 10: Control inputs in scenario A: v (solid black, in ms^{-1}), ω (dotted black, rads^{-1}) $\dot{\varphi}$ (dotted gray, rads^{-1}) and H (solid gray).

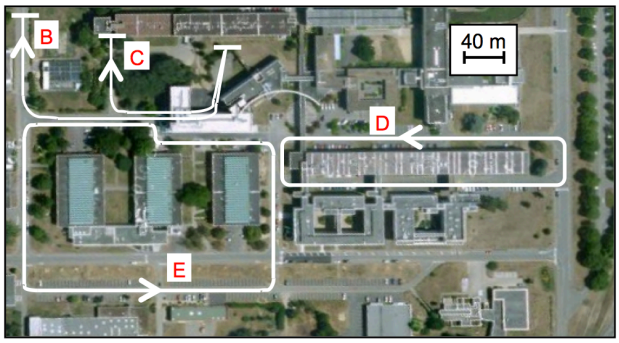


Figure 11: Map of the four navigation paths B, C, D, E.

the experiment. As the snapshots illustrate, 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 considered irrelevant, and do not deviate the robot from the path.

Another nice behaviour is shown in Fig. 12(center): if a stationing car is unavoidable, the robot decelerates and stops with (20), but, as soon as the car departs, the robot gradually accelerates (again with (20)), to resume navigation. In fact, as we mentioned in Section 4, when the best tentacle is clear up to distance δ_s, a high velocity can be applied: $v = v_s$, independently from the value of H. In the future, this feature of our approach could even be utilized for vehicle following.

An experiment with a crossing pedestrian is presented in Fig. 12(right). The pedestrian is considered irrelevant, until it enters the visual task tentacle (second image). Then, the clockwise tentacles are selected to pass between the person and the right side walk. When the path is clear again, the robot returns to the visual task tentacle, which is first counter-clockwise (fourth image) and then straight (fifth image).

In October 2011, as part of the final demonstration of the French ANR project CityVIP, we have validated our framework in a urban context, in the city of Clermont Ferrand. The experiments have taken place in the crowded neighbourhood of the central square Place de Jaude, shown in Fig. 13. For four entire days, our Cycab has navigated autonomously, continuously replaying a set of visual paths of up to 700 m each, amidst a variety of unpredictable obstacles, such as cars, pedestrians, bicycles and scooters. In Fig. 14, we show some significant snapshots of the experiments that were carried out in Clermont Ferrand. These include photos of the Cycab, as well as images acquired by the on-board camera during autonomous navigation. These experiments are also visible in the video shown in Extension #3.

In Fig. 14(a-c), Cycab is moving in a busy street, crowded with pedestrians and vehicles. First, in Fig. 14(a), we show a typical behaviour adopted for avoiding a crossing pedestrian: here, Cycab brakes as a lady with black skirt crosses the street. The robot would either stop or circumnavigate the person, and in

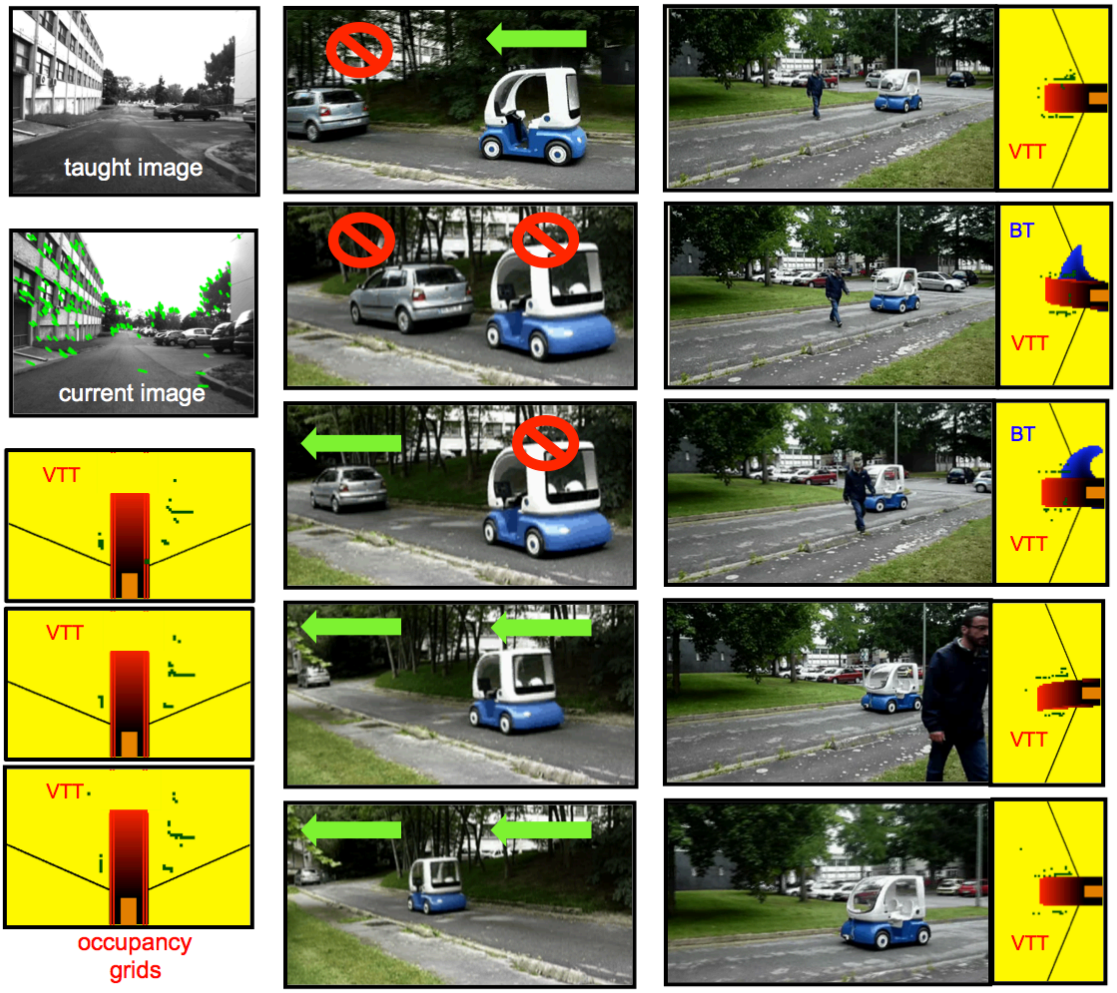


Figure 12: Validation with: irrelevant obstacles (left), traffic (center) and a moving pedestrian (right). The visual task tentacle and best tentacle (when different) are respectively indicated as VTT and BT in the occupancy grids.

four days no one has ever even closely been endangered nor touched by the vehicle. In many experiments, Cycab has navigated among fast moving vehicles (cars in Fig. 14(b), and a scooter in 14(c)), and manual security intervention was never necessary. The robot has also successfully avoided many fixed obstacles, including a stationing police patrol (Fig. 14(d)) and another electric vehicle (Fig. 14(e)). Obviously, when all visual features are occluded by an obstacle or lost, the robot stops.

Moreover, we have thoroughly tested the behaviour of our system with respect to varying light, which is an important aspect in outdoor appearance-based navigation. Varying light has been very common in the extensive Clermont Ferrand experiments, which would last the whole day, from the first light to sunset, both with cloudy and clear sky. In some experiments, we could control the robot in different lighting conditions,

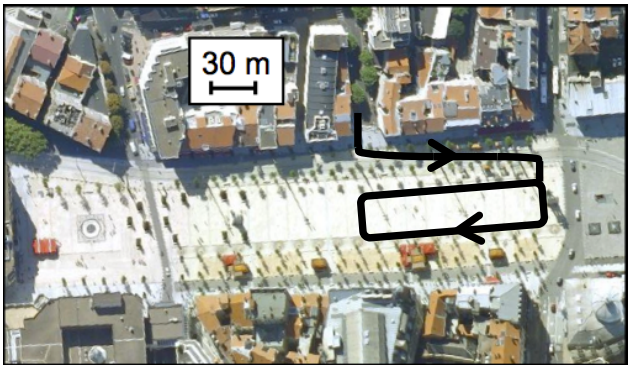


Figure 13: City center of Clermont Ferrand, with one of the navigation paths where the urban experiments have been carried out.

using the same taught database. For instance, Fig. 14(f) shows two images acquired approximately at the same position at 5 p.m. (top) and 11 a.m. (bottom), while navigating with the same key images. However, in spite of the robustness of the image processing algorithms. which has been proved in [Royer et al. 2007], in some cases (e.g., when the camera was overexposed to sunlight), the visual features required for navigation could not be detected. Future work in adapting the camera automatic shutter settings should solve this issue.

In the current version of our framework, moving obstacles are not specifically recognized and modelled. Although, as the experiments show, we are capable of avoiding slowly moving obstacles (e.g., crossing pedestrians or baby pushchairs as in Fig. 14(g)), the main objective of our future work will be to directly tackle this problem within the control law, in order to avoid fast obstacles as well. This can be done, for example, by estimating the velocity of the detected objects, and then using it to predict their future position. In our opinion, the main difficulty, in comparison with the case of static obstacles, will concern the accuracy and computation cost of this estimation process.

Overall, Cycab has navigated using an average of approximately sixty visual points on each image, and some paths have even been completed using less than 30 points. Along with all the cited technical aspects, the experiments highlighted the reactions of non-robotic persons to the use of autonomous ground vehicles in everyday life. Most passer-bys had not been informed of the experiments, and responded with curiosity, surprise, enthusiasm, and - rarely - slight apprehension.

7 Conclusions

A framework with simultaneous obstacle avoidance and outdoor vision-based navigation, without any 3D model or path planning has been presented. It merges a novel, reactive, tentacle-based technique with visual servoing, to guarantee path following, obstacle bypassing, and collision avoidance by deceleration.

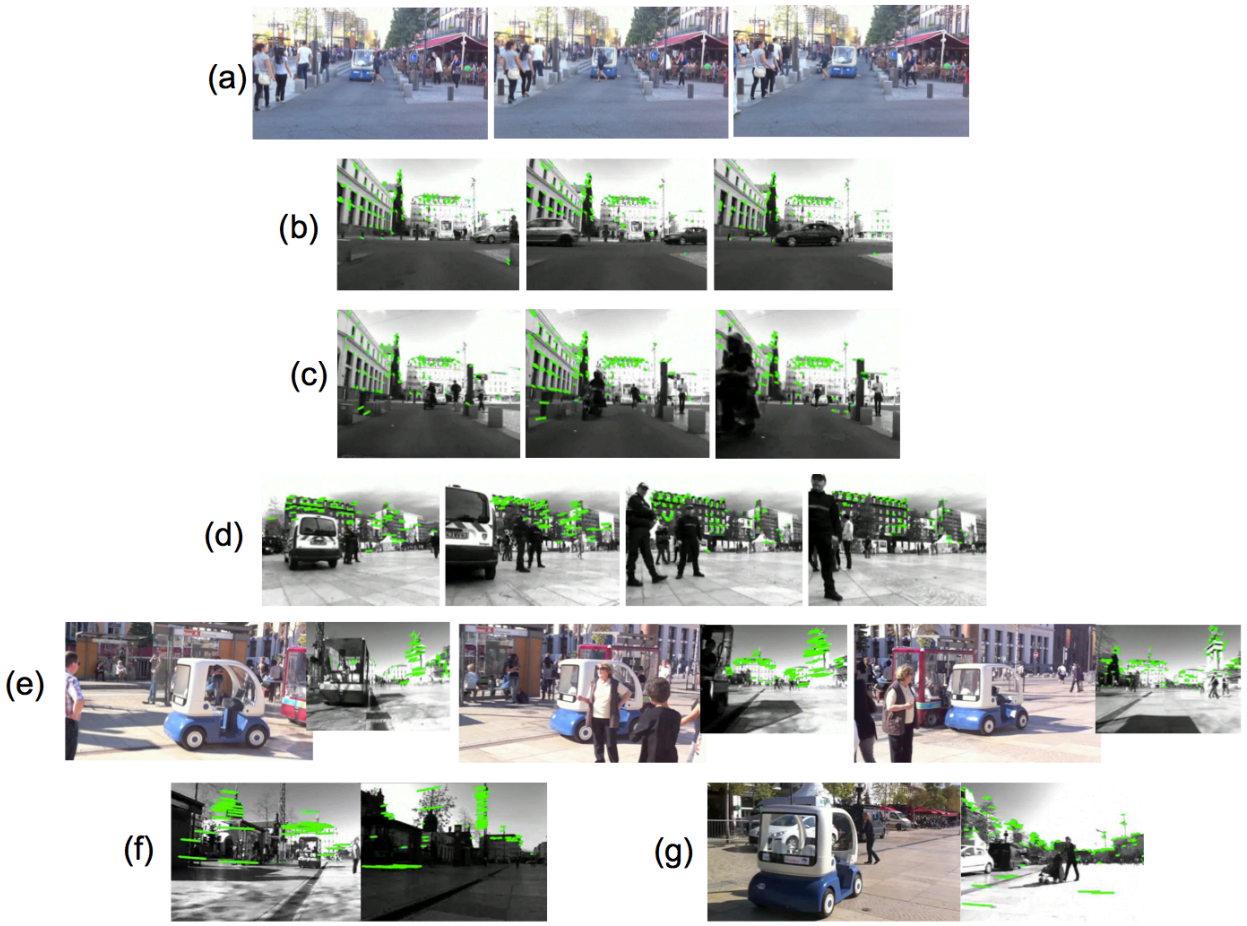


Figure 14: Snapshots of the urban experiments. (a) Avoiding a crossing pedestrian. (b-c) Navigating close to moving cars and to a scooter, respectively. (d-e) Avoiding a stationing police patrol and a stationing vehicle, respectively. (f) Navigating with different light conditions, using the same taught database. (g) Avoiding a pedestrian with a baby pushchair.

Since our method is purely sensor-based and pose-independent, it is perfectly suited for visual navigation. Extensive outdoor experiments, even in urban environments, show that it can be applied in realistic and challenging situations including moving obstacles (e.g., cars and pedestrians). To our knowledge, this is the first time that outdoor visual navigation with obstacle avoidance is carried out in urban environments at approximately 1 ms^{-1} on over 500 m, using neither GPS nor maps. In the near future, we plan to explicitly take into account the velocity of moving obstacles within our controller, in order to avoid fast obstacles, which are currently hard to deal with. Perspective work also includes automatic prevention of the visual occlusions provoked by the obstacles.

Appendix A: Index to Multimedia Extensions

The multimedia extensions (3 videos) to this article (see Table 2) are at www.ijrr.org. In all three videos, the segments linking the current and next key image points are drawn in light green in the current image. In the occupancy grid, the dangerous cell sets associated to the visual task tentacle and to the best tentacle (when different) are respectively shown in red and blue, and two black segments indicate the scanner amplitude. Only cells that can activate H (i.e., cells at distance $\Delta < \Delta_s$) have been drawn.

Table 2: Multimedia Extensions.

Extension	Media Type	Description
1	Video	Simulations of our navigation scheme in 6 different scenarios
2	Video	Experiments on our campus
3	Video	Extensive validation in the city center of Clermont Ferrand

Acknowledgment

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

References

[Allibert et al. 2008] G. Allibert, E. Courtial and Y. Touré, 2008, "Real-time visual predictive controller for image-based trajectory tracking of a mobile robot", *Int. Federation of Automatic Control World Congress*, Seoul, Korea.

[Becerra et al. 2010] H. M. Becerra, J. Courbon, Y. Mezouar and C. Sagüés, 2010, "Wheeled mobile robots navigation from a visual memory using wide field of view cameras", *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Taipei, Taiwan.

[Becerra et al. 2011] H. M. Becerra, G. López-Nicolás and C. Sagüés, 2011, "A Sliding-Mode-Control Law for Mobile Robots Based on Epipolar Visual Servoing From Three Views", *IEEE Trans. on Robotics*, vol. 27, no. 1, pp. 175–183.

[Bonin-Font et al. 2008] F. Bonin-Font, A. Ortiz and G. Oliver, 2008, "Visual navigation for mobile robots: a survey", *Journal of Intelligent and Robotic Systems*, vol. 53, no. 3, pp. 263–296.

[Bonnafous et al. 2001] D. Bonnafous, S. Lacroix and T. Siméon, 2001, "Motion generation for a rover on rough terrains", *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Maui, USA.

[Booij et al. 2007] O. Booij, B. Terwijn, Z. Zivkovic, B. Kröse, 2007, "Navigation using an appearance based topological map", *IEEE Int. Conf. on Robotics and Automation*, Rome, Italy.

[Broggi et al. 2010] A. Broggi, L. Bombini, S. Cattani, P. Cerri and R. I. Fedriga, 2010, "Sensing requirements for a 13000 km intercontinental autonomous drive", *IEEE Intelligent Vehicles Symposium*, San DIego, USA.

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

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

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

[Cherubini et al. 2009] A. Cherubini, M. Colafrancesco, G. Oriolo, L. Freda and F. Chaumette, 2009, "Comparing appearance-based controllers for nonholonomic navigation from a visual memory", *ICRA Workshop on safe navigation in open and dynamic environments*, Kobe, Japan.

[Cherubini and Chaumette 2011] A. Cherubini and F. Chaumette, 2011, "Visual navigation with obstacle avoidance", *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, San Francisco, USA.

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

[Courbon et al. 2009] J. Courbon, Y. Mezouar and P. Martinet, 2009, "Autonomous Navigation of Vehicles from a Visual Memory Using a Generic Camera Model", *IEEE Trans. on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 392–402.

[Davison et al. 2007] A. J. Davison, I. D. Reid, N. D. Molton and O. Stasse, 2007, "MonoSLAM: Real-Time Single Camera SLAM", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.29, no. 6, pp. 1052–1067.

[De Luca and Oriolo 1994] A. De Luca and G. Oriolo, 1994, “Local Incremental Planning for Nonholonomic Mobile Robots”, *IEEE Int. Conf. on Robotics and Automation*, San Diego, USA.

[De Luca et al. 2008] A. De Luca, G. Oriolo and P. Robuffo Giordano, 2008, “Feature Depth Observation for Image-based Visual Servoing: Theory and Experiments”, *The International Journal of Robotics Research*, vol. 27, no. 10, pp. 1093âĂŞ1116.

[Diosi et al. 2011] A. Diosi, S. Segvic, A. Remazeilles and F. Chaumette, 2011, “Experimental Evaluation of Autonomous Driving Based on Visual Memory and Image Based Visual Servoing”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 870-883.

[Durand et al. 2010] A. Durand Petiteville, M. Courdesses and V. Cadenat, 2010, “A new predictor/corrector pair to estimate the visual features depth during a vision-based navigation task in an unknown environment”, *International Conference on Informatics in Control, Automation and Robotics*, Rome, Italy.

[Elfes 1989] A. Elfes, “Using occupancy grids for mobile robot perception and navigation”, 1989, *Computer*, vol. 22(6), pp. 46-57.

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

[Fontanelli et al. 2009] D. Fontanelli, A. Danesi, F. A. W. Belo, P. Salaris and A. Bicchi, 2009, “Visual Servoing in the Large”, *The International Journal of Robotics Research*, vol. 28, no. 6, pp. 802−814.

[Goedemé et al. 2007] T. Goedemé, M. Nuttin, T. Tuytelaars and L. Van Gool, 2007, “Omnidirectional vision based topological navigation”, *Int. Journal of Computer Vision*, vol. 74, no. 3, pp. 219–236.

[Guerrero et al. 2008] J. J. Guerrero, A. C. Murillo and C. Sagüés, 2008, “Localization and Matching using the Planar Trifocal Tensor with Bearing-only Data”, *IEEE Trans. on Robotics*, vol. 24, no. 2, pp. 494–501.

[Harris and Stephens 1988] C. Harris and M. Stephens, 1988, “A combined corner and edge detector”, *4th Alvey Vision Conference*.

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

[Khatib 1985] O. Khatib, 1985, “Real-time obstacle avoidance for manipulators and mobile robots”, *IEEE Int. Conf. on Robotics and Automation*, Durham, UK.

[Lamiraux et al. 2004] F. Lamiraux, D. Bonnafous and O. Lefebvre, 2004, “Reactive Path Deformation for Nonholonomic Mobile Robots”, in *IEEE Trans. on Robotics*, vol. 20, no. 6, pp. 967–977.

[Lapierre et al. 2007] L. Lapierre, R. Zapata and P. Lepinay, 2007, “Simultaneous path following and obstacle avoidance control of a unicycle-type robot”, *IEEE Int. Conf. on Robotics and Automation*, Rome, Italy.

[Latombe 1991] J. C. Latombe, 1991, “Robot motion planning”, 1991, *Kluwer Academic, Dordredt*.

[Lee et al. 2010] T.-S. Lee, G.-H. Eoh, J. Kim and B.-H. Lee, 2010, “Mobile robot navigation with reactive free space estimation”, *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Taipei, Taiwan.

[López-Nicolás et al. 2010] G. López-Nicolás, N. R. Gans, S. Bhattacharya, C. Sagüés, J. J. Guerrero and S. Hutchinson, 2010, “An Optimal Homography-Based Control Scheme for Mobile Robots with Nonholonomic and Field-of-View Constraints”, *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, vol. 40, no. 4, pp. 1115–1127.

[López-Nicolás and Sagüés 2011] G. López-Nicolás and C. Sagüés, 2011, “Vision-based exponential stabilization of mobile robots”, *Autonomous Robots*, vol. 30, no. 2, pp. 293−306.

[Lowe 2004] D. G. Lowe, 2004, “Distinctive image features from scale-invariant keypoints”, *Int. Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110.

[Mariottini et al. 2007] G. L. Mariottini, G. Oriolo, D. Prattichizzo, 2007, “Image-Based Visual Servoing for Nonholonomic Mobile Robots Using Epipolar Geometry”, *IEEE Trans. on Robotics*, vol. 23, no. 1, pp. 87–100.

[Minguez et al. 2008] J. Minguez, F. Lamiraux and J. P. Laumond, 2008, “Motion planning and obstacle avoidance”, in *Springer Handbook of Robotics*, B. Siciliano, O. Khatib (Eds.), Springer, pp. 827–852.

[Nunes et al. 2009] U. Nunes, C. Laugier and M. Trivedi, 2009, “Introducing Perception, Planning, and Navigation for Intelligent Vehicles” in *IEEE Trans. on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 375–379.

[Ohya et al. 2008] A. Ohya, A. Kosaka and A. Kak, 1998, "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, pp. 969–978.

[Quinlan and Khatib 1993] S. Quinlan and O. Khatib, 1993, "Elastic bands: connecting path planning and control", *IEEE Int. Conf. on Robotics and Automation*, Atlanta, USA.

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

[Scaramuzza and Siegwart 2008] D. Scaramuzza and R. Siegwart, 2008, "Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles", *IEEE Trans. on Robotics*, vol. 24, no. 5, pp. 1015–1026.

[Sciavicco and Siciliano 2000] L. Sciavicco and B. Siciliano, 2000, "Modeling and Control of Robot Manipulators", *Springer*.

[Šegvić et al. 2008] S. Šegvić, A. Remazeilles, A. Diosi and F. Chaumette, 2008, "A mapping and localization framework for scalable appearance-based navigation", *Computer Vision and Image Understanding*, vol. 113, no. 2, pp. 172–187.

[Shi and Tomasi 1994] J. Shi and C. Tomasi, 1994, "Good features to track", *IEEE Conf. on Computer Vision and Pattern Recognition*, Seattle, USA.

[Tahri and Chaumette 2005] O. Tahri and F. Chaumette, 2005, "Point-based and region-based image moments for visual servoing of planar objects", *IEEE Trans. on Robotics*, vol. 21, no. 6, pp. 1116–1127.

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

[Wada et al. 2009] T. Wada, S. Doi and S. Hiraoka, 2009, "A deceleration control method of automobile for collision avoidance based on driver's perceptual risk", *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.

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

[Zhang et al. 2008] Y. Zhang, W. C. Lin and Y.-K. S.Chin, 2008, "Driving skill characterization: A feasibility study", *IEEE Int. Conf. on Robotics and Automation*, Pasadena, USA.

[Zhang and Kleeman 2009] A. M. Zhang and L. Kleeman, 2009, "Robust Appearance Based Visual Route Following for Navigation in Large-scale Outdoor Environments", *Int. Journal of Robotics Research*, vol. 28, no. 3, pp. 331−356.