

Autonomous Visual Navigation and Laser-based Moving Obstacle Avoidance

Andrea Cherubini, Fabien Spindler, François Chaumette

▶ **To cite this version:**

Andrea Cherubini, Fabien Spindler, François Chaumette. Autonomous Visual Navigation and Laser-based Moving Obstacle Avoidance. IEEE Transactions on Intelligent Transportation Systems, 2014, 15 (5), pp.2101-2110. 10.1109/TITS.2014.2308977 . hal-00954360

HAL Id: hal-00954360

https://inria.hal.science/hal-00954360

Submitted on 1 Mar 2014

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.

Autonomous Visual Navigation and Laser-based Moving Obstacle Avoidance

Andrea Cherubini, *Member, IEEE,* Fabien Spindler, and François Chaumette, *Fellow, IEEE*

***Abstract*—Moving obstacle avoidance is a fundamental requirement for any robot operating in real environments, where pedestrians, bicycles and cars are present. In this paper, we propose and validate a framework for avoiding moving obstacles during visual navigation with 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, our robot is able to avoid static as well as moving obstacles, which were not present during teaching, and which are sensed by an on-board lidar. The proposed approach takes explicitly into account obstacle velocities, estimated using an appropriate Kalman-based observer. The velocities are then used to predict the obstacle positions within a tentacle-based approach. Finally, our approach is validated in a series of real outdoor experiments, showing that when the obstacle velocities are considered, the robot behaviour is safer, smoother, and faster than when it is not.**

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

I. INTRODUCTION

One of the main objectives of recent robotics research is the development of vehicles capable of autonomously navigating in unknown environments [3], [4], [5]. The success of the DARPA Urban Challenges [6] has heightened expectations that autonomous cars will soon be able to operate in environments of realistic complexity. In this field, information from visual sensors is often used for localization [7], [8] or for navigation purposes [9], [10], [11].

Nevertheless, a critical issue for successful navigation remains motion safety, especially when the robot size and dynamics make it potentially harmful. Hence, an important task is obstacle avoidance, which consists of either generating a collision-free trajectory to the goal, or of decelerating to prevent collision whenever bypassing is impossible [12]. Obstacle avoidance has traditionally been handled by two techniques [13]: the deliberative approach [14], usually consisting of a motion planner, and the reactive approach [15], based on the instantaneous sensed information.

The task that we focus on is outdoor visual navigation: a wheeled vehicle, equipped with an actuated pinhole camera and with a forward-looking lidar, must follow a path represented by key images, without colliding. In the past, obstacle avoidance has been integrated in visual navigation [16], [17] and path following [18], [19], 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 reactive (i.e., merely sensor-based) solution, which does not need a global model of the environment and trajectory. Moreover, the proposed solution copes with moving obstacles, which are common in dynamic, real-world environments.

Reactive strategies include: the vector field histogram [20], obstacle-restriction method [21], and closest gap [22]. In our previous work [2], we presented a novel sensor-based method guaranteeing that obstacle avoidance had no effect on visual navigation. However, in that work, we did not consider moving obstacles. In the literature, researchers have taken into account the obstacle velocities to deal with this issue. We hereby survey the main papers dedicated to moving obstacle avoidance.

The approach presented in [23] is one of the first where static and moving obstacles are avoided, based on their current positions and velocities relative to the robot. The maneuvers are generated by selecting robot velocities outside of the *velocity obstacles*, that would provoke a collision at some future time. Planning in the velocity space makes it possible to consider the robot dynamics. This paradigm has been adapted in [24] to the car-like robot kinematic model, and extended in [25] to take into account unpredictably moving obstacles. This has been done by using reachability sets to find matching constraints in the velocity space, called Velocity Obstacle Sets.

Another pioneer method that has inspired many others is the Dynamic Window [26], that is derived directly from the dynamics of the robot, and is especially designed to deal with constrained velocities and accelerations. The method consists of two steps: first, a valid subset of the control space is generated, and then an optimal solution (driving the robot with maximum obstacle clearance) is sought within it. A generalization of the dynamic map that accounts for moving obstacle velocities and shapes is presented in [27], where a union of polygonal zones corresponding to the non admissible velocities controls the robot, and prevents collisions. In [28], the Dynamic Window has been integrated in a graph search algorithm for path planning, to drive the robot trajectories within a global path. A planning approach is also used in [29], where the likelihood of obstacle positions is input to a Rapidly-exploring Random Tree algorithm.

In [30], motion safety is characterized by three criteria, respectively related to the model of the robotic system, to the model of the environment and to the decision making process. The author proves that motion safety cannot be guaranteed in the presence of moving objects (i.e., the robot may inevitably

A. Cherubini is 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.

This paper has been published in part in [1] and in [2].

Manuscript received XXX; revised XXX.

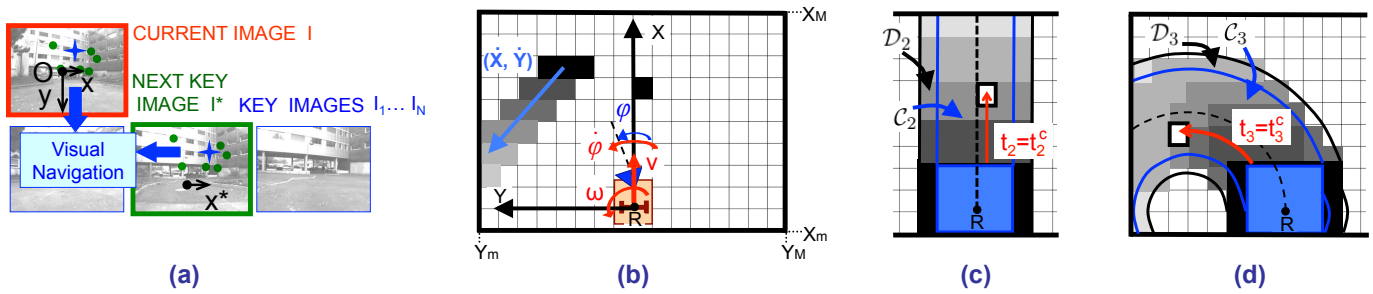


Fig. 1. (a) Current and next key images, and key image database. (b) Top view of the robot, with actuated camera and control inputs v, ω and $\dot{\varphi}$. A static (right) and moving (left) object are observed (black); we show the object velocity $\left(\dot{X}, \dot{Y}\right)$, and future occupied cells c_i in grey, increasingly light with t_{i0}. (c, d) Tentacles (dashed black), with classification areas, corresponding boxes and delimiting arcs of circle, and cells $\mathbf{c}_i \in \mathcal{D}_j$ displayed in grey, increasingly light with increasing t_{ij}.

collide at some time in the future). More recently [31], the same researchers have defined the Braking Inevitable Collision States (BICS) as states such that, whatever the future braking trajectory, a collision will occur. In that paper, the BICS are used to achieve passive motion safety.

Although all these approaches have proved effective, none of them deals with moving obstacle avoidance during visual navigation. In [2], we presented a framework that guarantees that obstacle avoidance has no effect on the visual task. In the present paper, we further improve that framework, by designing a reactive approach that can deal with moving obstacles as well. Our approach is based on tentacles [32], i.e. candidate trajectories (arcs of circles) that are evaluated during navigation, both for assessing the context, and for designing the task in case of danger. The main contribution of this paper is the improvement of that framework, to take into account the obstacle velocities. First, we have designed a Kalman-based observer for estimating the obstacle velocities. Then, we have adapted the tentacles designed in [2], to effectively take into account these velocities. Finally, we experimentally validate our approach in a series of experiments.

The article is organized as follows. In Section II, all the relevant variables are defined. In Section III and IV, we explain respectively how the obstacle velocities are estimated, and how they are used to predict possible collisions. Then, in Section V, the control law from [2] is recalled, and adapted to deal with moving obstacles. Experimental results are reported in Section VI, and summarized in Section VII.

II. PROBLEM DEFINITION

This section is, in part, taken from [2]. Referring to Fig. 1(a, b), 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). The robot control inputs are:

$$\mathbf{u} = [v,\ \omega,\ \dot{\varphi}]^{\top}.$$

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

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(a)) is selected. Then, during autonomous navigation, the current image, noted I, is compared with the next key image $I^* \in \{I_1, \ldots, I_N\}$, and a relative pose estimation between I and I^* is used to check when the robot passes the pose where I^* was acquired. For key image selection, and visual point detection and tracking, we use the algorithm proposed in [33]. The output of this algorithm, which is used by our controller, is the set of points visible both in I and I^*. Then, navigation consists of driving the robot forward, while I is driven to I^*. We maximize similarity between I and I^* using only the abscissa x of the centroid of points matched on I and I^* to control the robot heading. If no points can be matched between I and I^*, the robot stops. This can typically occur in the presence of an occluding obstacle; however, navigation is resumed as soon as the obstacle moves and the features are again visible. When I^* 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 lidar in a plane parallel to the ground. For obstacle modeling, we use the occupancy grid presented in Fig. 1(b): 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. An appropriate choice for $|X_m|$ is the length of the robot, since obstacles behind cannot hit it as it advances. Any grid cell $\mathbf{c} = [X, Y]^{\top}$ is considered currently occupied (black in Fig. 1(b)) if an obstacle has been sensed there. For cells lying in the lidar area, only the current lidar reading is considered. For the other cells, we use past readings, displaced with odometry. The set of occupied cells with their estimated velocities, denoted by $\mathcal{O}$, is used, along with the robot geometric and kinematic characteristics, to derive possible future collisions. This approach is different from the one in [2], where only the *currently* occupied cells in the grid were considered. To estimate the obstacle velocities, and therefore update $\mathcal{O}$, we have designed an obstacle observer, detailed just below.

III. OBSTACLE OBSERVER

The detection and tracking of obstacles is crucial for collision-free navigation. Of particular interest are potentially dynamic objects (i.e., objects that could move) since their presence and potential change of state will influence the planning of actions and trajectories. Obviously, estimating the velocity of these objects is fundamental.

Compared to areas where known road network information can provide background separation, unknown environments present a more challenging scenario, due to low signal to noise ratio. Recent works [34], [35] have tackled these issues. In [34], classes of interest for autonomous driving (i.e., cars, pedestrians and bicycles) are identified, using shape information and a RANSAC-based edge selection algorithm. The authors of [35] apply a foreground model that incorporates geometric as well as temporal cues; then, moving vehicles are tracked using a particle filter. Both works rely on the Velodyne HDL-64E S2, a laser range finder that provides rich 3D point clouds, to classify moving obstacles. Instead, we target solutions based uniquely on a 2D lidar, and we are not interested in recognizing the object classes.

In practice, we base our work on two assumptions:

1) All objects are rigid.
2) The instantaneous curvature of the object trajectories (i.e., the ratio between their angular and translational velocities) is small enough to assume that their motion is purely translational over short time intervals. Hence, the translational velocities of all points on an object are identical and equal to that of its centroid.

Both assumptions are plausible for the projection on the ground of walls, most vehicles and even pedestrians.

Under these assumptions, for each object, the state to be estimated will be composed of the coordinates of its centroid in $\mathcal{F}_{\mathcal{R}}$, and by their derivatives[1]:

$$\mathbf{x} = \left[X, Y, \dot{X}, \dot{Y}\right]^{\top}.$$

Using a first-order Markov model (which is plausible for low object accelerations), the state at time t is evolved from the state at $t - \Delta t$ (Δt is the sampling time) according to:

$$\mathbf{x}(t) = \mathbf{F}\mathbf{x}(t - \Delta t) + \mathbf{w}(t), \tag{1}$$

where $\mathbf{w}(t) \sim N(0, \mathbf{Q})$ is assumed to be Gaussian white noise, with covariance $\mathbf{Q}$ and, the state transition model is:

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

At time t, an observation $\mathbf{z}(t)$ of the object centroid coordinates is derived from lidar data. It is related to the state by:

$$\mathbf{z}(t) = \mathbf{H}\mathbf{x}(t) + \mathbf{v}(t), \tag{2}$$

where $\mathbf{v}(t) \sim N(0, \mathbf{R})$ is assumed to be Gaussian white noise with covariance $\mathbf{R}$, and the observation model is:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

Let us outline the steps of our recursive algorithm for deriving $\hat{\mathbf{x}}(t)$ (our estimate of $\mathbf{x}(t)$), based on current observations $\mathbf{z}(t)$, and on previous states $\mathbf{x}(t - \Delta t)$.

1) At time t, all currently occupied cells in $\mathcal{O}$ are clustered in objects, using a threshold on pairwise cell distance, and the current observation of the centroid coordinates $\mathbf{z}(t)$ is derived for each object.
2) All of the object centroids that have been observed at some time in the recent past (we look back in the last $2s$) are displaced by odometry, to derive their coordinates $X(t - \Delta t)$ and $Y(t - \Delta t)$.
3) The observed and previous object centroids (outputs of steps 1 and 2) are pairwise matched according to their distance[2]. We then discern between three cases:
   - For matched objects, the previous centroid velocity is obtained by numerical differentiation, and input to a Kalman filter, based on equations (1-2), and on the outputs of step 2, to derive $\hat{\mathbf{x}}(t)$.
   - For unmatched objects currently observed, we set:
   $$\hat{\mathbf{x}}(t) = \begin{bmatrix} \mathbf{z}(t) \\ 0 \\ 0 \end{bmatrix}.$$
   - For unmatched unobserved objects, the centroid coordinates are memorized (these will be the inputs for step 2).

The output of our algorithm is, at each iteration t, the estimate of the object centroid coordinates and of its velocities:

$$\hat{\mathbf{x}}(t) = \left[\hat{X}(t), \hat{Y}(t), \hat{\dot{X}}(t), \hat{\dot{Y}}(t)\right]^{\top}.$$

Then, each currently occupied cell $\mathbf{c}_i$ is associated to the estimated velocity of the object it belongs to, or to null velocity, if it has not been associated to any object. Set $\mathcal{O}$ is finally formed by all the occupied cell states:

$$\begin{bmatrix} \mathbf{c}_i \\ \dot{\mathbf{c}}_i \end{bmatrix}^{\top} \in \mathcal{O},$$

that encode the cell current coordinates and velocities in the robot frame.

Let us briefly discuss the particular case of a group of near moving obstacles. If these are close, they are clustered into a single object. If one or more obstacles leave the group, these will be treated as new objects, and their estimated velocity is immediately reset, since there are no matches in the past. The choice of restarting the observer in this case is reasonable, since to leave the group, these obstacles had to substantially vary their velocity.

[1] If assumption 2 is not met, the orientation and angular velocity of the object must be added to the state vector $\mathbf{x}$, that is be estimated by our observer.

[2] High obstacle velocity can hinder this step: if the ratio between the obstacle velocity relative to the vehicle, and the obstacle processing algorithm framerate is strong, the obstacle centroid position in the grid will strongly vary between successive iterations, making matching impossible.

In the next section, we will show how $\mathcal{O}$ is used to predict possible collisions, and accordingly adapt the control strategy. We will assess the danger of each cell by considering the time that the robot will navigate before eventually colliding with it. Without loss of generality, in the next section this time is measured from the current instant t.

IV. OBSTACLE MODELLING

A. Obstacle occupation times

At this stage, the trajectory of each occupied cell in $\mathcal{O}$ can be predicted to evaluate possible collisions with the robot. More concretely, we will just estimate the times at which each cell in the grid will be - eventually - occupied by an obstacle. We assume that velocities of all occupied cells in $\mathcal{O}$ remain constant over time horizon T. This is a plausible assumption, since the estimations of the obstacles positions and velocities are updated at every iteration, by the approach outlined above. Then, for each $\mathbf{c}_i$ that may be occupied by an obstacle within T, we can predict initial

$$t_{i0}\left(\mathbf{c}_i, \mathcal{O}\right) \in [0, T]$$

and final

$$t_{if}\left(\mathbf{c}_i, \mathcal{O}\right) \in [t_{i0}, T]$$

obstacle occupation times, as a function of the set of occupied cell states $\mathcal{O}$. For cells occupied by a static object and belonging to $\mathcal{O}$, we obtain $t_0 = 0$ and $t_f = T$. For cells that will not be occupied within time T, we set $t_0 = t_f = \infty$. Examples of a static (1 cell) and moving (3 cells) object are shown in Fig. 1(b), with future occupied cells $\mathbf{c}_i$ displayed in grey, increasingly light with increasing t_{i0}. Below, we explain how the cell occupation times t_0 and t_f will be used to check collisions with the possible robot trajectories.

B. Tentacles

As in [2], we use a set of drivable paths (tentacles), both for perception and motion execution. Each tentacle j is a semicircle 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:

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

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 Fig. 1(c, d), 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 two classification areas (*dangerous* and *collision*), which are obtained by rigidly displacing, along the tentacle, two rectangular boxes, with decreasing size, both overestimated with respect to the real robot dimensions. For each tentacle j, the sets of cells belonging to the two classification areas (shown in Fig. 1) are noted $\mathcal{D}_j$ and $\mathcal{C}_j \subset \mathcal{D}_j$. As we will show below, the largest classification area $\mathcal{D}$ will be used to select the safest tentacle, while the thinnest one $\mathcal{C}$ determines the eventual necessary deceleration.

In summary, the tentacles exploit the robot geometric and kinematic characteristics. Specifically, the robot geometry (i.e., the vehicle encumbrance) defines the two classification areas $\mathcal{C}$ and $\mathcal{D}$, 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 Section V, to choose the best tentacle and to eventually slow down or stop the robot.

C. Robot occupation times

For each dangerous cell in tentacle j, i.e., for each cell $\mathbf{c}_i \in \mathcal{D}_j$), we compute the *robot occupation time* t_{ij}. This is an estimate of the time at which the large box will enter the cell, assuming the robot follows the tentacle at the current velocity. To calculate t_{ij}, we assume that the robot motion is uniform, and displace the box at the current robot linear velocity v, and at angular velocity $\omega_j = \kappa_j v$. We can then calculate robot occupation time t_{ij}:

$$t_{ij}\left(\mathbf{c}_i, v, \kappa_j\right) \in I\!R^+.$$

For instance, if the robot is not moving ($v = 0$), for every tentacle j, the cells on the box will have $t_{ij} = 0$, and all other cells in $\mathcal{D}_j$ will have $t_{ij} = \infty$. Also note that for a given cell, t_i may differ according to the tentacle that is considered. In Fig. 1(c, d), the cells $\mathbf{c}_i \in \mathcal{D}_j$ have been displayed in grey, increasingly light with increasing t_{ij}.

D. Dangerous and collision instants

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

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

and

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

In both cases, we seek the earliest time at which a cell is simultaneously occupied by the obstacle and by the robot box. Assuming constant robot and obstacle velocities, these metrics give an approximation of the time that the robot can travel along the tentacle without colliding. Obviously, overestimating the bounding boxes size leads also to more conservative values of t_j and t_j^c. In the following, we explain how these metrics are used: in particular, with t_j we assess the

danger on each tentacle to decide whether to follow it or not, while t_j^c determines if the robot should decelerate on tentacle j. Computation of t_j and t_j^c is illustrated, for $j = \{2, 3\}$, in the example of Fig. 1.

E. Tentacle risk function

The danger on each tentacle is assessed by *tentacle risk function* H_j. This scalar function is derived from the tentacle dangerous instant, and will be used by the controller as explained in Section V. We use t_j and tuned thresholds $t_d > 0$ and $t_s > t_d$ (d stands for dangerous, and s for safe), to design the tentacle risk function:

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

Note that H_j smoothly varies from 0, when possible collisions are in the far future, to 1, when they are forthcoming. If $H_j = 0$, the tentacle is tagged as *clear*. All the H_j are compared (with a strategy explained below), to determine H in (3) and select the *best tentacle* for navigation.

V. CONTROL SCHEME

In our control scheme, 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. The rotation makes the robot follow the *best* tentacle in $\mathcal{K}$, which is selected using the strategy explained below. 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* $H \in [0, 1]$, also defined below.

Stability of the desired tasks has been guaranteed in [2] by:

$$\begin{cases} v = (1 - H)\, v_s + H v_u \\ \omega = (1 - H)\, \frac{\lambda_x (x^* - x) - j_v v_s + \lambda_\varphi j_{\dot{\varphi}} \varphi}{j_\omega} + H \kappa_b v_u \\ \dot{\varphi} = H\, \frac{\lambda_x (x^* - x) - (j_v + j_\omega \kappa_b) v_u}{j_{\dot{\varphi}}} - (1 - H)\, \lambda_\varphi \varphi \end{cases} \tag{3}$$

In the above equations:

- H is the risk function on the best tentacle: $H = H_b$; hence, it is null if and only if the best tentacle is clear.
- $v_s > 0$ is the translational velocity in the *safe context* (i.e., when $H = 0$). It must be maximal on straight path portions, and smoothly decrease when the features quickly move in the image, i.e., at sharp robot turns (large ω), and when the camera pan angle φ is strong. Hence, we define v_s as:

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

with function σ defined as:

$$\sigma(\omega, \varphi) = [1 + \tanh(\pi - k_\omega |\omega|)]\, [1 + \tanh(\pi - k_\varphi |\varphi|)]\,.$$

Function (4) 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 determined by empirically tuned positive parameters k_ω and k_φ.

- $v_u \in [0, v_s]$ is the translational velocity in the *unsafe context* ($H = 1$). It is designed as:

$$v_u(\delta_b) = \begin{cases} v_s & \text{if } t_b^c \geq t_s^c \\ v_s \sqrt{t_b^c - t_d^c / t_s^c - t_d^c} & \text{if } t_d^c < t_b^c < t_s^c \\ 0 & \text{if } t_b^c \leq t_d^c \end{cases} \tag{5}$$

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

- x and x^* 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^* and of φ to 0.
- j_v, j_ω and $j_{\dot{\varphi}}$ are the components of the Jacobian relating $\dot{x}$ and $\mathbf{u}$. They are:

$$\begin{aligned} j_v &= \tfrac{-\sin\varphi + x\cos\varphi}{\zeta} \\ j_\omega &= \tfrac{\rho(\cos\varphi + x\sin\varphi)}{\zeta} + 1 + x^2 \\ j_{\dot{\varphi}} &= 1 + x^2, \end{aligned}$$

where ρ is the abscissa of the optical center in the robot frame $\mathcal{F}_\mathcal{R}$, and ζ is the feature centroid depth with respect to the camera.

- κ_b is the curvature of the best tentacle. Here we detail how such a tentacle is determined. Initially, we calculate the path curvature that the robot would follow if $H = 0$:

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

In [2], we proved that κ is always well-defined, i.e., that $j_\omega \neq 0$. We constrain κ to the interval of feasible curvatures $[-\kappa_M, \kappa_M]$, and derive its two neighbors in $\mathcal{K}$: κ_n and κ_{nn}. Let κ_n be the nearest one, denoted as the *visual tentacle*[3]. Its situation risk function H_v is obtained by linear interpolation of the neighbours:

$$H_v = \frac{(H_{nn} - H_n)\, \kappa + H_n \kappa_{nn} - H_{nn} \kappa_n}{\kappa_{nn} - \kappa_n}. \tag{6}$$

If $H_v = 0$, the visual tentacle is clear and can be followed: we set $\kappa_b = \kappa_n$. Instead, if $H_v \neq 0$, we seek a clear tentacle ($H_j = 0$). First, we search among the tentacles between the visual task one and the best one at the previous iteration[4], noted κ_{pb}. If many are present, the closest to the visual tentacle is chosen. If none of the tentacles within $[\kappa_n, \kappa_{pb}]$ is clear, we search among the others. If no tentacle in $\mathcal{K}$ is clear, the one with minimum H_j is chosen. Ambiguities are again solved first with the distance from κ_n, then from κ_{nn}.

Let us shortly recall the main features of (3), which are detailed in [2]. When $H = 0$ (i.e., if the 2 neighbour tentacles

[3]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 $\kappa_n > \kappa_{nn}$.

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

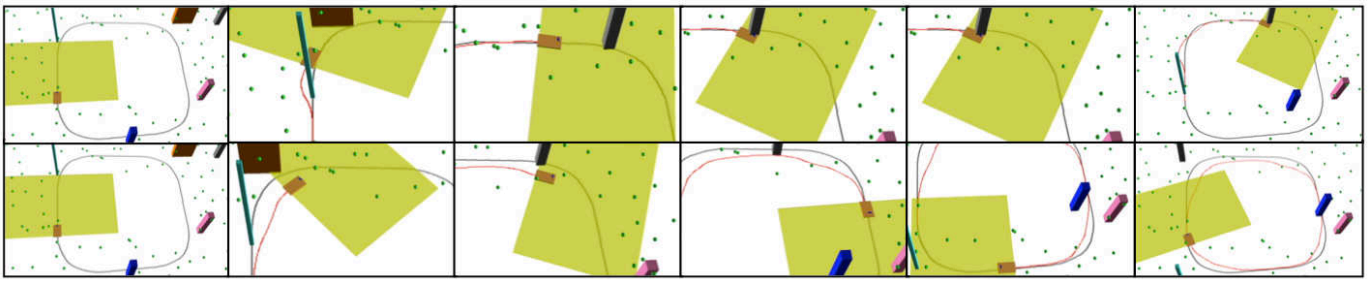


Fig. 2. Six steps of the simulations: the taught path must be followed by the robot with methods S (top) and M (bottom) and 4 moving and 1 static obstacles. Visual features (spheres), occupancy grid, and replayed paths are also shown.

are clear), 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 ($\varphi = 0$). When $H = 1$, $\dot{\varphi}$ ensures the visual task, and the two other inputs guarantee that the best tentacle is followed: $\omega/v = \kappa_b$. In general ($H \in [0, 1]$), the robot navigates between the taught and the best paths, and a high velocity v_s can be applied if the path is clear for future time t_s^c.

VI. EXPERIMENTS

In this section, we will detail the experiments that were used to validate our approach. These are also shown in the video attached to this paper.

All experiments have been carried out on our CyCab vehicle, set in car-like mode (i.e., using the front wheels for steering). The maximum speed attainable by the CyCab is 1.3 ms^{-1}. For preliminary simulations, we made use of Webots[5], where we designed a virtual CyCab, and distributed random visual features, represented by spheres, as well as physical obstacles, in the environment. 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 (3). 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 (3) is sent to the robot as soon as the visual information is available (10 Hz).

The occupancy grid is built by projecting the laser readings from the two 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. For the situation risk function, we use $t_s = 6$ s and $t_d = 4.5$ s, for the unsafe translational velocity, we use $t_s^c = 5$ s, and $t_d^c = 2$ s, and as control gains: $\lambda_x = 1$ and $\lambda_\varphi = 0.5$. We set $\kappa_M = 0.35$ m^{-1} (the Cycab maximum applicable curvature, corresponding to a minimum turn radius of 2.86 m). All these values were tuned after a few simulations, and proved appropriate throughout the experiments. 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 (6)), all the cells in the dangerous area $\mathcal{D}$ 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. The experiments showed that the computational cost of laser processing, using the chosen number of 21 tentacles, was never a critical issue with respect to that of image processing.

In all the experiments, at first, the robot is driven along a taught path. Then, moving and static objects are present on the path, while the robot replays it to follow the key images. To evaluate the experiments, we verify if the robot is able to complete the taught path until the final key image (this was the case in all experiments), and we measure its linear velocity v, averaged over the whole experiment, and denoted $\bar{v}$. We do not consider the 3D pose error with respect to the taught path, since our task is defined in the image space, and not in the pose space. Besides, some portions of the replayed paths, corresponding to the obstacle avoidances, are far from the taught ones. However, these deviations are indispensable to avoid collisions. In some experiments, we also compare the new approach that is presented here, and that takes into account the obstacle velocities, with the original one designed in [2]. We denote these respectively as approach M and S (for Mobile and Static).

Let us firstly describe the simulations, shown in Fig. 2. The taught visual path is a closed clockwise loop of $N = 20$ key images, and the robot must replay it, while avoiding 4 moving obstacles, with velocity norms up to 1 ms^{-1}, and a static one. Higher obstacle velocities are difficult to estimate due to the low frequency of laser processing (12.5 Hz). However, it is noteworthy to point out that 1 ms^{-1} is the walking speed of a quick pedestrian. With approach M, the vehicle is able to follow the whole path without colliding, whereas when S is used, the robot collides with the third obstacle. Let us now detail the robot behaviour in the two cases. The first obstacle (a box moving straight towards the robot) is avoided by both approaches, although with M motion prediction leads to a smoother and earlier circumnavigation. With M, the robot is faster, and reaches the brown box while it is crossing its way; but since the box is expected to leave, the robot just waits for the path to return free. With S, the robot arrives at the same point late, when the box is far. The third, grey box moves straight towards the robot, like the first one. Since it is slightly faster, this time S is not reactive enough, and a

[5]www.cyberbotics.com

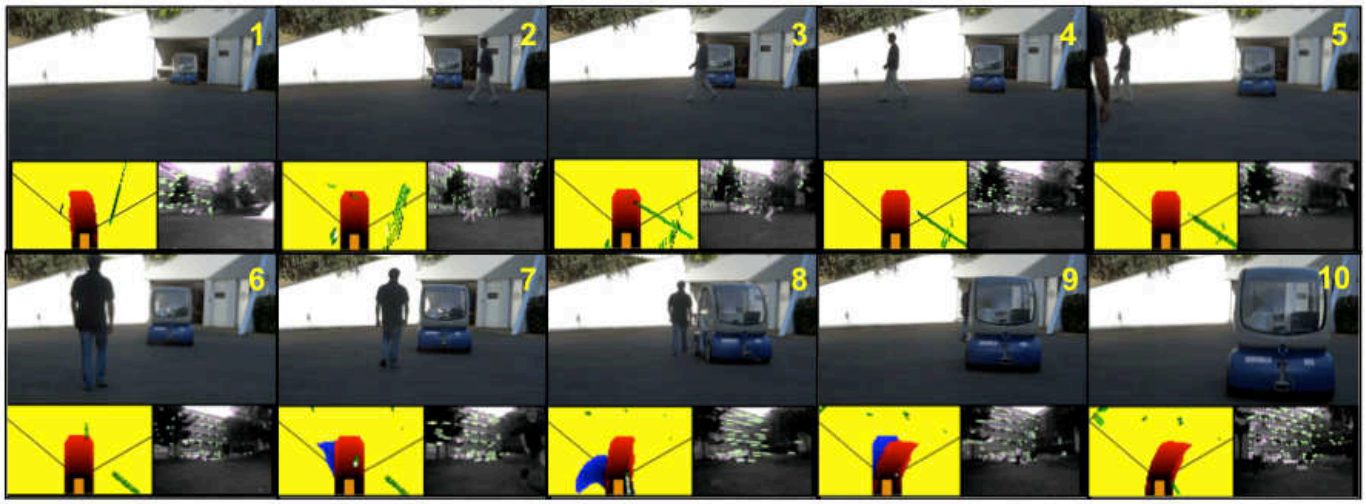


Fig. 3. Ten relevant iterations of the experiment with two crossing pedestrians. For each iteration, we show the occupancy grid (left) and current image (right). In the occupancy grid, the dangerous cell sets associated with the visual tentacle and to the best tentacle (when different) are shown, and two black segments indicate the lidar amplitude. Only cells that we predict to be occupied in the next T s have been drawn in the grid. The segments link the current and next key image points.



Fig. 4. Comparison between methods S (top) and M (bottom) as a pedestrian crosses the path in front of the robot.

collision occurs. On the other hand, with M the boxes are easily avoided. The new approach also prevails in speed: the average velocity $\bar{v} = 0.67$ ms^{-1} with M, and $\bar{v} = 0.49$ ms^{-1} with S.

After the simulations, the framework has been ported on our CyCab vehicle.

In a first experiment, two pedestrians are passing during navigation: one crosses the path, and the other walks straight towards the robot. We show, in Fig. 3, relevant iterations with the corresponding occupancy grids and currently viewed images. In the occupancy grid, the propagation of cells occupied by the persons is visible at iterations 2-8. With the crossing pedestrian (iterations 2-5), since no collision is predicted, the robot keeps following the visual tentacle. Instead, with the forward walking pedestrian, a collision is predicted at iteration 7; then, the robot selects the best tentacle to avoid the person. Visual path replaying is again successful, with $\bar{v} = 0.87$ ms^{-1}.

Then, we have compared methods S and M in an experiment, where a single pedestrian crosses the taught path in front of the robot (see Fig. 4, with control inputs plotted in Fig. 5). The smooth trend of $\mathbf{u}$ at the beginning and end of

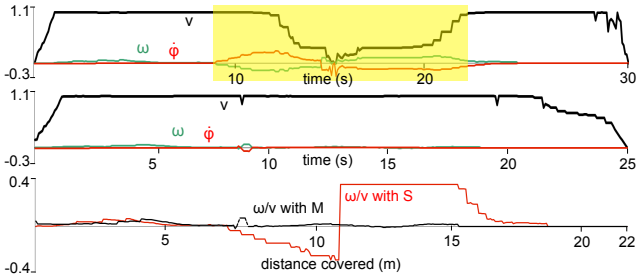


Fig. 5. Single pedestrian experiment. Top and center, respectively: control inputs using S and M, with v (ms^{-1}), ω (rads^{-1}), $\dot{\varphi}$ (rads^{-1}), and iterations with strong H highlighted. Bottom: applied curvature ω/v (in m^{-1}) using S and M.

the experiments is due to the acceleration saturation carried out at the CyCab low-level control. With controller S (top in both figures), the robot attempts avoidance on the right, since tentacles on the left are occupied by the person. This is clearly a doomed strategy, which leads the robot toward the pedestrian. Then, the robot must decelerate and almost stop ($v \approx 0$ after 15 s) when the pedestrian is very near.

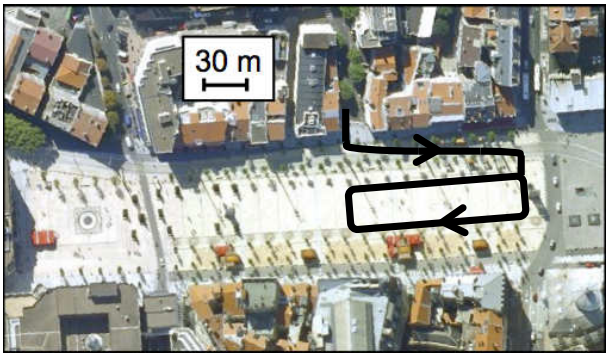


Fig. 6. City center of Clermont Ferrand, with one of the navigation paths where the urban experiments have been carried out.

Navigation is resumed only once the path is clear again. On the other hand, with controller M, as the pedestrian walks, the prediction of his future position makes him irrelevant from a safety viewpoint: risk function H (highlighted in Fig. 5), which was relevant with S, is now null. Hence, the robot does not need to reduce its speed ($\bar{v}$ is 0.89 ms^{-1} with M, and 0.76 ms^{-1} with S) nor to deviate from the path (in the bottom of Fig. 5, the applied curvature is smaller). The image error with respect to the database $x - x^*$, averaged over the experiment is also reduced with M: 7 instead of 12 pixels.

As part of the final demonstration of the French ANR project CityVIP, we have also 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. 6. 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. 7, 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 shown in the video attached to this paper.

In Fig. 7(a-c), Cycab is moving in a busy street, crowded with pedestrians and vehicles. First, in Fig. 7(a), we show the behaviour adopted in the presence of a pedestrian (a lady with black skirt) crossing a narrow street: the Cycab brakes, since avoidance is impossible. In general, the robot would either circumnavigate the person or stop, and in 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. 7(b), and a scooter in 7(c)), and manual security intervention was never necessary. The robot has also successfully avoided many static obstacles, including a stationing police patrol (Fig. 7(d)) and another electric vehicle (Fig. 7(e)). When all visual features are occluded by an obstacle, the robot stops, but resumes navigation as soon as the obstacle moves and the features are again visible.

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, using the same taught database. For instance, Fig. 7(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 [33], 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.

Overall, Cycab has navigated using an average of approximately 60 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.

VII. CONCLUSIONS

We presented a novel framework with simultaneous laser-based moving obstacle avoidance and outdoor vision-based navigation, without any 3D model or path planning. It merges a reactive, tentacle-based technique with visual servoing, to guarantee path following, obstacle bypassing, and collision avoidance by deceleration. In particular, for the first time obstacle velocities are accounted for within a visual navigation scheme. To estimate the obstacle velocities, we have designed a Kalman-based observer. Then, we utilize the velocities to predict possible collisions between robot and obstacles. Our approach is validated in a series of experiments (including urban environments), and it is compared with a similar controller that does not consider obstacle velocities.

The results show that, by predicting the obstacle displacements within the candidate tentacles, the robot behaviour is safer and smoother, and higher velocities can be attained. The framework can be applied in realistic and challenging situations including real moving obstacles (e.g., cars and pedestrians). To our knowledge, this is the first time that outdoor visual navigation with moving obstacle avoidance is carried out in urban environments at approximately 1 ms^{-1} on over 500 m, using neither GPS nor maps.

In the future, we will investigate scenarios, where obstacles are not translating, as assumed here, and can approach the vehicle from behind. For the latter case, the current configuration (forward-looking lidar) must be modified. Perspective work also includes automatic prevention of the visual occlusions provoked by the obstacles.

REFERENCES

[1] A. Cherubini, B. Grechanichenko, F. Spindler and F. Chaumette. "Avoiding Moving Obstacles during Visual Navigation", *IEEE Int. Conf. on Robotics and Automation*, 2013.

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

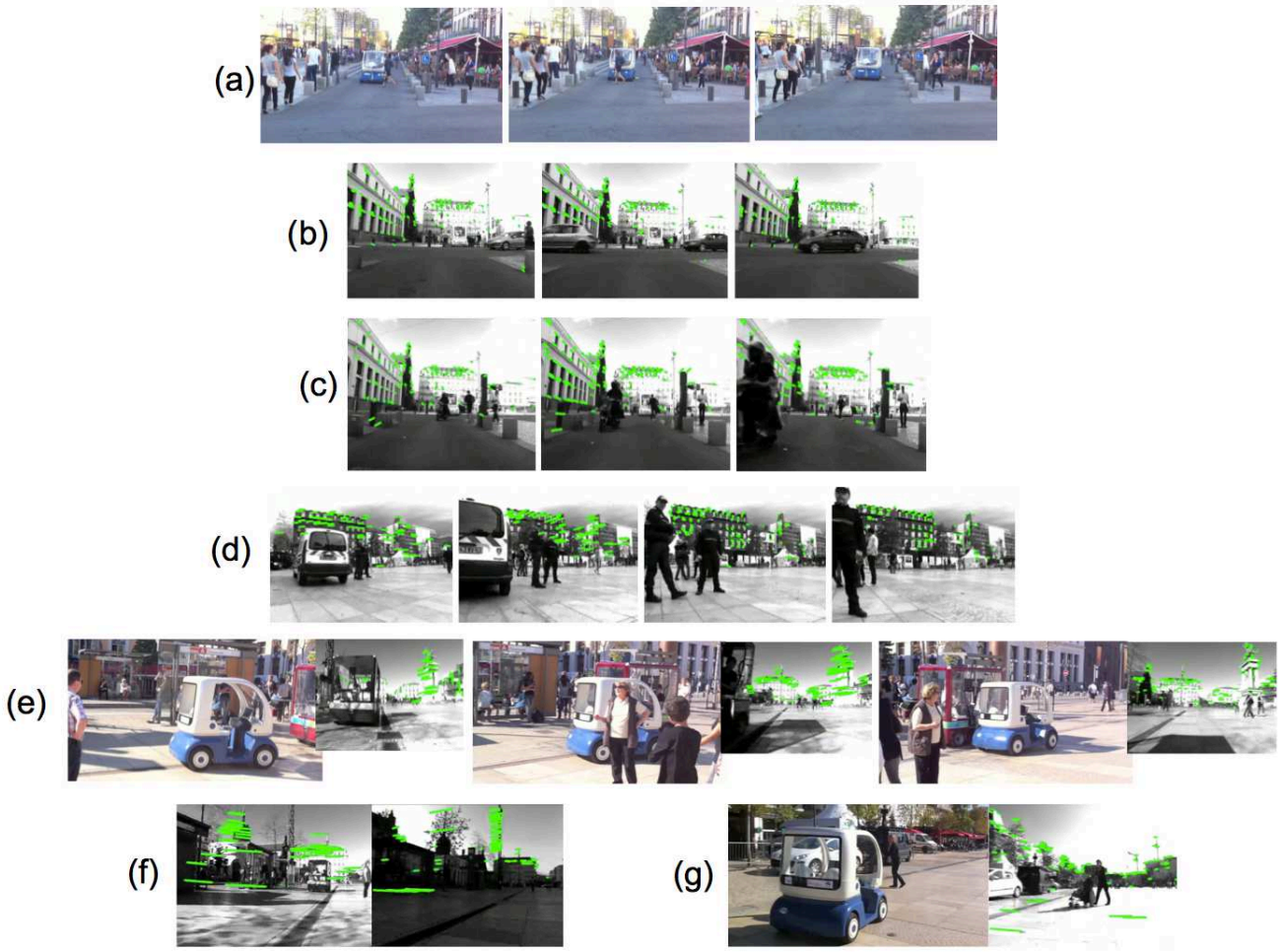


Fig. 7. Snapshots of the urban experiments. (a) Avoiding a crossing pedestrian. (b-c) Navigating close to moving cars and to a scooter, respectively. (d-e) Avoiding a stationing police patrol and a stationing vehicle, respectively. (f) Navigating with different light conditions, using the same taught database. (g) Avoiding a pedestrian with a baby pushchair.

[3] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. Van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian and P. Mahoney, "Stanley: The robot that won the DARPA Grand Challenge" in *Journal of Field Robotics*, vol. 23, no. 9, 2006, pp. 661 - 692.

[4] 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.

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

[6] 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.

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

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

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

[10] G. López-Nicolás, N. R. Gans, S. Bhattacharya, C. Sagüés, J. J. Guerrero and S. Hutchinson, "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, 2010, pp. 1115–1127.

[11] A. Diosi, S. Segvic, A. Remazeilles and F. Chaumette, "Experimental Evaluation of Autonomous Driving Based on Visual Memory and Image Based Visual Servoing", *IEEE Trans. on Intelligent Transportation Systems*, vol. 12, no. 3, 2011, pp. 870–883.

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

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

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

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

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.
[21] J. Minguez, "The Obstacle-Restriction Method (ORM) for robot obstacle avoidance in difficult environments", *IROS*, 2005.
[22] M. Mujahad, D. Fischer, B. Mertsching and H. Jaddu "Closest Gap based (CG) reactive obstacle avoidance navigation for highly cluttered environments", *IROS*, 2010.
[23] P. Fiorini and Z. Shiller, "Motion Planning in Dynamic Environments Using Velocity Obstacles", in *Int. Journal of Robotics Research*, vol. 17, no. 7, 1998, pp. 760 – 772.
[24] D. Wilkie and J. Van den Berg, D. Manocha, "Generalized Velocity Obstacles", *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009.
[25] A. Wu and J P. How, "Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles", in *Autonomous Robots* vol. 32, no. 3, 2012, pp. 227-242.
[26] D. Fox, W. Burgard and S. Thrun, "The Dynamic Window approach to obstacle avoidance", in *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, 1997, pp. 23–33.
[27] B. Damas and J. Santos-Victor, "Avoiding Moving Obstacles: the Forbidden Velocity Map", *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009.
[28] M. Seder and I. Petrović, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles", *IEEE Int. Conf. on Robotics and Automation*, 2007.
[29] C. Fulgenzi, A. Spalanzani, C. Laugier, "Probabilistic motion planning among moving obstacles following typical motion patterns", in *IEEE/RSJ Int. Conf. on Intelligent RObots and Systems*, 2009.
[30] T. Fraichard, "A Short Paper about Motion Safety", *IEEE Int. Conf. on Robotics and Automation*, 2007.
[31] S. Bouraine, T. Fraichard and H. Salhi, "Provably safe navigation for mobile robots with limited field-of-views in dynamic environments", in *Autonomous Robots*, vol. 32, no. 3, 2012, pp. 267-283.
[32] 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.
[33] E. Royer, M. Lhuillier, M. Dhome and J.-M. Lavest, "Monocular vision for mobile robot localization and autonomous navigation", in *Int. Journal of Computer Vision*, vol. 74, no. 3, 2007, pp. 237–260.
[34] D. Zeng Wang, I. Posner and P. Newman, "What Could Move? Finding Cars, Pedestrians and Bicyclists in 3D Laser Data", in *IEEE Int. Conf. on Robotics and Automation*, 2012.
[35] N. Wojke and M. Häselich, "Moving Vehicle Detection and Tracking in Unstructured Environments", in *IEEE Int. Conf. on Robotics and Automation*, 2012.



Andrea Cherubini Andrea Cherubini received the MSc in Mechanical Engineering in 2001 from the University of Rome La Sapienza and a second MSc in Control Systems in 2003 from the University of Sheffield, U.K. In 2008, he received the PhD degree in Control Systems from the Dipartimento di Informatica e Sistemistica, University of Rome La Sapienza . From 2008 to 2011, he worked as post-doctoral fellow at Inria Rennes. He is currently Associate Professor at Université de Montpellier 2, and researcher at LIRMM. His research interests include: visual servoing for mobile robots, nonholonomic robot navigation, human-robot interaction, and humanoid robotics.



Fabien Spindler Fabien Spindler graduated from ENI Brest engineer school (specialisation in electronics) in 1992, and received the Master of Science degree in Electronics and Aerospace Telecommunication from Supaero in Toulouse in 1993. Since 1994, he has been with Inria in Rennes as research engineer. He is in charge of the material and software management of several robotic experimentation platforms dedicated to researches in visual servoing. His interests include software engineering for the design of real-time computer vision and robotics applications. He is the software architect of the open-source ViSP (Visual Servoing Platform) library and is involved in the software transfer to industrial or academic partners.



François Chaumette François Chaumette was graduated from École Nationale Supérieure de Mécanique, Nantes, France, in 1987. He received the Ph.D. degree in computer science from the University of Rennes, France, in 1990. Since 1990, he has been with Inria in Rennes where he is now Senior Research Scientist and head of the Lagadic group (`http://www.irisa.fr/lagadic`). His research interests include robotics and computer vision, especially visual servoing and active perception. Dr. Chaumette is IEEE Fellow. He received the AFCET/CNRS Prize for the best French thesis in automatic control in 1991. He also received with Ezio Malis the 2002 King-Sun Fu Memorial Best IEEE Transactions on Robotics and Automation Paper Award. He has been Associate Editor of the IEEE Transactions on Robotics from 2001 to 2005 and is now in the Editorial Board of the Int. Journal of Robotics Research.