HAL
open science

Stand-alone obstacle avoidance controller for image-based navigation

Adrien Durand-Petiteville, Viviane Cadenat, Thierry Sentenac

To cite this version:

Adrien Durand-Petiteville, Viviane Cadenat, Thierry Sentenac. Stand-alone obstacle avoidance controller for image-based navigation. IEEE LARS 2022 / SBR 2022 -19th IEEE Latin American Robotics Symposium / 14th Brazilian Symposium on Robotics, Oct 2022, São Paulo, Brazil. p. 133-138, 10.1109/LARS/SBR/WRE56824.2022.9995933 . hal-03807426

HAL Id: hal-03807426

https://hal.science/hal-03807426v1

Submitted on 9 Oct 2022

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.

Stand-alone obstacle avoidance controller for image-based navigation

A. Durand-Petiteville
Mechanical Engineering Department
Universidade Federal de Pernambuco
Recife, Brazil
adrien.durandpetiteville[at]ufpe.br

V. Cadenat
LAAS-CNRS
Université Paul Sabatier Toulouse III
Toulouse, France
cadenat[at]laas.fr

T. Sentenac
LAAS-CNRS and Institut Clément Ader (ICA)
IMT Mines Albi, INSA, UPS, ISAE
Albi, France
sentenac[at]laas.fr

***Abstract*—In this paper we present a stand-alone reactive avoidance controller inspired by the spirals described by insects during their flights. Unlike state-of-the-art obstacle avoidance methods, it only focuses on circumventing hazardous elements and can be coupled to navigation controllers working in different state space, such as the image one. We describe the obstacle avoidance problem by following a set of spirals, the spiral parameters computation to deal with convex/concave obstacles, and the spiral following controller design. Finally, a set of simulations obtained with ROS/Gazebo show the efficiency of the proposed approach to navigate in a cluttered environment.**

I. Introduction

This paper addresses the obstacle avoidance problem during an image-based navigation in an unknown environment. For this work, the motion planning techniques where a collision-free trajectory to the target is computed, leading to a global solution to the navigation problem, are discarded. Indeed, they consider a perfect model of the robot and of the environment, and fail when the surroundings are unknown and unpredictable [1]. For this reason, it is proposed to focus on the obstacle avoidance techniques. They consist in driving the robot towards the goal relying on the data collected by the sensors during the navigation execution [1]. Thus, these techniques, classified as reactive, introduce the sensor information within the control loop, and allow adapting the motion to any contingency incompatible with initial plans at the cost of a local solution. Among the most representative obstacle avoidance methods, let us mention first the ones computing the motion in one step. For example, the Potential Fields method [2] which is based on physical analogies and directly reduces the sensor information to motion control. Then, one focuses on approaches requiring more than one step: first, an intermediate set of motion controls is computed, and next, one of them is chosen. The intermediate set might be a subset of motion directions for methods such as the Vector Field Histogram (VFH) [3] or Obstacle Restriction [4], or a subset of velocity controls in the case of the Dynamic Window Approach (DWA) [5] or Velocities Obstacles (VO) [6] ones. However, all these methods share a common approach: they compute a motion leading to the goal and avoiding obstacles, and to do so they require to know the desired pose in the metric space. Thus, when the goal is expressed in a different space, such as the image space when controlling a robot via Visual Servoing [7], it is not possible to rely on one of these approaches to avoid obstacles. To tackle this issue, it was proposed in [8] to design a tentacle-based obstacle avoidance method coupled with an Image-Based Visual Servoing (IBVS). It relies on an occupancy grid which is then used to evaluate the collision risk of a set of tentacles. When the risk of collision is too high, the robot follows the tentacle with the best evaluation.

In this work, we present a navigation framework inspired by the behavior-based systems [9], where the navigation task is divided into two sub-tasks: reaching the desired pose and avoiding obstacles. The proposed system is then composed of two controllers and one finite-state machine activating one of the them based on the current situation. The first controller is an IBVS driving the robot towards a desired pose expressed in the image space, while the second one is a spiral follower using data provided by a laser range finder in order to perform the obstacle avoidance. The choice of IBVS is motivated by the relatively low cost of cameras and their ability to detect landmarks in challenging environments. Moreover, this approach does not require the robot to estimate its metric pose on a global frame. Regarding the obstacle avoidance controller, its design is inspired by the work presented in [10] and modeling the spirals described by insects during their flights. This work has already been used in [11] to avoid aircraft, in [12] and [13] to perform a u-turn in an orchard, and in [14] to avoid convex obstacles during a metric-based navigation. In this work, it is proposed to use the spiral model to design a stand-alone obstacle avoidance controller that can be coupled with any navigation controller, such as an IBVS one. The model is simple enough to provide a computationally efficient implementation, while being sufficiently versatile to lead to advance paths circumventing complex static obstacles. Thus, in this paper we present (i) the approach allowing to use the spiral model to avoid convex and concave obstacles, (ii) the design of a time-varying spiral following controller, and (iii) the finite-state machine coupling the controllers.

The rest of the paper is organized as follows. The next

This work was supported by FACEPE (APQ-0139-3.04/20) and ANR (ANR-20-CE33-001-01)

978-1-6654-6280-8/22/$31.00 ©2022 IEEE

section is dedicated to a succinct presentation of the IBVS approached used in our navigation framework. Then, the obstacle avoidance approach is described, including the overall description, the data processing step, the design of the controller and the presentation of the finite-state machine. Finally, results obtained via simulation are presented to show the relevancy and efficiency of the proposed approach.

II. Preliminaries

A. System modeling

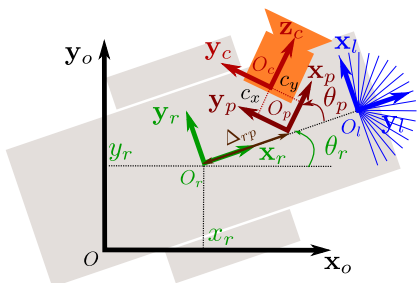


Fig. 1: Differential drive robot model.

In this work we aim at controlling a differential robot equipped with a camera embedded on a pan-platform and a laser rangefinder attached to the mobile base. To model the robotic system, the following frames are introduced as shown in Figure 1: $\mathbf{F}_o(O, \mathbf{x}_o, \mathbf{y}_o, \mathbf{z}_o)$ as the world frame, $\mathbf{F}_r(O_r, \mathbf{x}_r, \mathbf{y}_r, \mathbf{z}_r)$ as the robot frame, $\mathbf{F}_p(O_p, \mathbf{x}_p, \mathbf{y}_p, \mathbf{z}_p)$ as the pan-platform frame, $\mathbf{F}_c(O_c, \mathbf{x}_c, \mathbf{y}_c, \mathbf{z}_c)$ as the camera frame, and $\mathbf{F}_l(O_l, \mathbf{x}_l, \mathbf{y}_l, \mathbf{z}_l)$ as the laser frame. The robot pose is given by $\chi_r = [x_r, y_r, \theta_r, \theta_p]^T$, where x_r and y_r are the coordinates of O_r in $\mathbf{F}_o$, and θ_r and θ_p are respectively the robot orientation with respect to $\mathbf{x}_o$, and the pan-platform orientation with respect to $\mathbf{x}_r$. Finally, $\mathbf{q} = [\upsilon, \omega_r, \omega_{pp}]^T$ denotes the system control input vector, where υ and ω_r are the mobile base linear and angular velocities, and ω_{pp} is the pan-platform angular velocity with respect to $\mathbf{F}_r$.

As the camera is mounted on the pan-platform of a differential drive robot, its motion is controlled via the robotic system, and its number of degrees of freedom is reduced to three (two translation motions along $\mathbf{y_c}$ and $\mathbf{z_c}$, one rotation motion around $\mathbf{x_c}$ as shown in Fig. 1). Thus, the camera kinematic screw $\nu_\mathbf{c}$ can be written as follows [15]:

$$\nu_\mathbf{c} = \mathbf{J}\mathbf{q} = \begin{bmatrix} -\sin(\theta_p) & \Delta_{rp}\cos(\theta_p) + c_x & c_x \\ \cos(\theta_p) & \Delta_{rp}\sin(\theta_p) - c_y & -c_y \\ 0 & -1 & -1 \end{bmatrix} \mathbf{q} \quad (1)$$

where $\mathbf{J}$ is the robot Jacobian, c_x and c_y are the coordinates of O_c along axes $\mathbf{x}_p$ and $\mathbf{y}_p$, and Δ_{rp} is the distance between O_r and O_p (see Figure 1).

B. The vision-based positioning task

The navigation task is made of two sub-tasks: the vision-based positioning sub-task, detailed hereafter, and the obstacle avoidance one, presented in the following section. The vision-based positioning task consists in controlling the robot to make the camera reach a pose defined in the image space. To do so, we rely on a visual features vector $\mathbf{s}$, which values depend on the relative camera pose with respect to a reference static landmark. The landmark is characterized by n_{vf} interest points whose coordinates in $\mathbf{F}_c$ are defined by (x_j, y_j, z_j) with $j \in [1, ..., n_{vf}]$. Their projections on the image plane are defined as $\mathbf{s_j} = (X_j, Y_j)$ (see Fig. 2). Thus, the visual features vector $\mathbf{s}$ is given by:

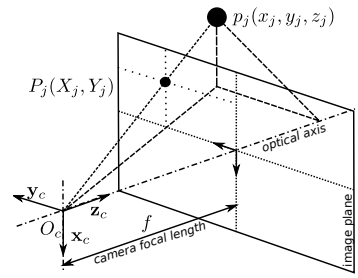


Fig. 2: Perspective camera model.

$$\mathbf{s} = [X_1, Y_1, ..., X_{n_{vf}}, Y_{n_{vf}}]^T \quad (2)$$

For the considered robot and visual features, the relation between $\dot{s}$ and $\mathbf{q}$ is given by [7]: $\dot{\mathbf{s}} = \mathbf{L}\nu_\mathbf{c} = \mathbf{L}\mathbf{J}\mathbf{q}$ where $\mathbf{L} = [\mathbf{L}_1^T, ..., \mathbf{L}_{n_{vf}}^T]^T$ is the so-called interaction matrix with $\mathbf{L}_j$ defined by [7]:

$$\mathbf{L}_j = \begin{bmatrix} 0 & \frac{X_j}{z_j} & \frac{X_j Y_j}{f} \\ \frac{f}{z_j} & \frac{Y_j}{z_j} & (f + \frac{Y_j^2}{f}) \end{bmatrix} \quad (3)$$

with f the focal distance. The visual servoing task will be successfully performed if $\mathbf{s}$ converge towards $\mathbf{s}^*$, that is if the error function $\mathbf{e}_{vs} = \mathbf{s} - \mathbf{s}^*$ vanishes. To do so, we apply the visual servoing technique given in [16] to mobile robots as in [17]. It consists in zeroing $\mathbf{e}_{vs}$ by imposing an exponential decrease, that is: $\dot{\mathbf{e}}_{vs} = -\lambda_{vs}\mathbf{e}_{vs}$ where λ_{vs} is a positive scalar or a positive definite matrix which fixes the decrease rate. The corresponding control law is given by:

$$\mathbf{q}_{vs} = -\lambda_{vs}(\mathbf{L}\mathbf{J})^+(\mathbf{s} - \mathbf{s}^*) \quad (4)$$

where $(\mathbf{L}\mathbf{J})^+$ is the pseudo-inverse of $\mathbf{L}\mathbf{J}$.

III. Obstacle avoidance

A. The obstacle avoidance strategy

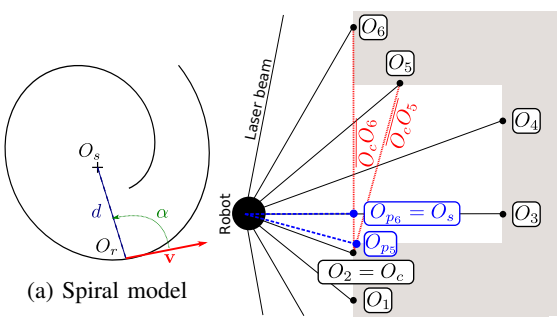
In this work, we design an obstacle avoidance strategy relying on the spirals described by insects during their flights. To do so, we use the spiral model presented in [10] (see Fig. 3a). Let us first consider a point O_r moving according to the velocity vector $\mathbf{v}$. O_r represents the insect position or the robot frame origin in our case. Let consider a second point O_s, static, whose polar coordinates in $\mathbf{F}_r$ can be expressed as $\mathbf{O_s} = (\alpha, d)$, where α is the oriented angle from $\mathbf{v}$ to the $\mathbf{O_r O_s}$ vector, and d the distance between O_r and O_s. It is shown in [10] that if both $\mathbf{v}$ and α are constant during a time

interval Δt, then O_r describes a spiral whose center is O_s during Δt. In such a case, the following equation holds:

$$\dot{d} = -v\cos(\alpha) \tag{5}$$

Thus, (5) shows that the type of spiral only depends on α. Indeed, if $\alpha \in]-\pi; 0[$, then O_r turns clockwise with respect to O_s and counter-clockwise if $\alpha \in]0; \pi[$. Moreover, if $\alpha \in]-\pi; -\frac{\pi}{2}[\cup]\frac{\pi}{2}; \pi[$, then the spiral is outward and inward if $\alpha \in]-\frac{\pi}{2}; 0[\cup]0; \frac{\pi}{2}[$. It becomes a circle if $\alpha = \pm\frac{\pi}{2}$, with a radius equal to d. Finally, if $\alpha = 0$, then O_r follows a straight line towards O_s or away from it if $\alpha = \pi$.

The spiral model being presented, we now describe the obstacle avoidance strategy. Each new data are processed to determine if an obstacle should be considered as dangerous (see Section III-B). In such a case, we determine a reference spiral the robot has to follow to avoid a collision. It consists in calculating a center point O_s and an angle α^*, both guaranteeing non-collision. We then use a spiral following controller (see Section III-C) to avoid the obstacle. It should be noted that a reference spiral is computed for each new laser data. Thus, our method does not seek to follow a unique spiral over time, and the robot does not describe a spiral-shaped path. On the contrary, it follows a different spiral at each iteration, allowing thus to generate complex path and to deal with challenging obstacles.



(a) Spiral model

(b) O_s computation

Fig. 3: Obstacle avoidance models

B. Determination of the reference spiral parameters

To compute the reference spiral parameters, the laser point cloud is first preprocessed to extract the data corresponding to the obstacle to avoid. Thus, it is divided into n_o subsets, corresponding to the n_o obstacles in the sensor field of view. To do so, the range values set is browsed and a new subset is created and/or closed when the difference between two consecutive ranges is larger than a user-defined threshold Δ_o. Next, obstacles that are too close to allow the robot to pass between them are merged. Thus, if the distance between two points of two different subsets is smaller than a user-defined threshold Δ_m, the subsets are merged. Δ_m is chosen based on the robot diameter and a minimal safety distance. Finally, among the subsets, we select the closest one to the robot as the point cloud representing the obstacle to avoid. Note, that it could represent a unique physical obstacle, *e.g.,* a sofa, or several close obstacles, *e.g.,* the legs of a chair.

We now focus on the reference spiral center O_s. We first compute O_c, the closest obstacle point to the robot. Next, we compute an approximation of the closest point lying on a segment line closing a concavity (see Fig. 3b). To do so, we first calculate O_{p_i}, the orthogonal projection of the robot center onto the line passing by O_c and O_i, for each point O_i of the point cloud. If O_{p_i} belongs to the line segment $\overline{O_c O_i}$, it becomes a candidate for O_s. Once O_{p_i} has been calculated for each O_i, we select among them and O_c the closest one to the robot as O_s. Thus, the spiral center is either O_c or a point belonging to the segment line closing the concavity.

Regarding the spiral angle, it is proposed to define a time-varying reference angle $\alpha^*(t)$. Indeed, if the error to minimize only depends on a constant reference angle α^*, then the distance to the spiral center, *i.e.,* to the obstacle, is not controlled and depends on the initial condition. If the controller is activated far away from the obstacle, the initial distance is then large. By following such a spiral the robot safely avoids the obstacle but describes an inefficient path in regards of the navigation task. On the contrary, if activated close to the obstacle, the initial distance is small. It leads to an efficient path but the robot might collide with the obstacle, especially during the transient phase. For a navigation, we would like to start the avoidance as soon as the obstacle is considered dangerous, while passing at the smallest possible distance from it to obtain an efficient path. For this reason, the robot has to follow a spiral at a given distance from the its center, *i.e.,* to control both α and d. We then define the following time-varying reference angle $\alpha^*(t)$:

$$\alpha^*(t) = \alpha_c + \alpha_d \epsilon(t) \tag{6}$$

$\alpha^*(t)$ is composed of two parts. The first one, α_c, is constant and equal to $\pm\pi/2$ to maintain a constant robot-obstacle distance. The second one, $\alpha_d\epsilon(t)$, makes the robot reach the desired distance. Thus, when $d > d^*$, $\alpha^*(t)$ should tend towards 0, which is the best value to get the robot closer to the spiral center. On the contrary, when $d < d^*$, $\alpha^*(t)$ should tend towards $\pm\pi$, which is the best value to get the robot away from the spiral center. To do so, α_d represents the maximal value that can be added/subtracted to α_c to make the robot converge towards the spiral without changing its circumvention direction. Then $\epsilon(t)$ should have its norm equal to 1 when $||d(t) - d^*|| \geq ||d(0) - d^*||$, and equal to 0 when $d(t) = d^*$. To do so, α_d and $\epsilon(t)$ are defined as:

$$\alpha_d = \begin{cases} \alpha_c & \text{if } d(0) > d^* \\ sign(\alpha_c) * \pi - \alpha_c & \text{if } d(0) < d^* \end{cases} \tag{7}$$

$$\epsilon(t) = \text{sign}(d^* - d(t)) \min\left(\left|\frac{d^* - d(t)}{|d^* - d(0)|}\right|, 1 \right) \tag{8}$$

First, (7) guarantees that the sense of rotation is not modified. Indeed, we obtain $\alpha^* \in [0, \pi]$ when $\alpha_c \in [0, \pi]$, and $\alpha^* \in [0, -\pi]$ when $\alpha_c \in [0, -\pi]$. Next, (8) computes $\epsilon(t)$ as the normalized error between d^* and $d(t)$ and makes it saturated

to ± 1. Thus, its value belongs to the domain $[0, 1]$ if $d(0) < d^*$ or $[-1, 0]$ if $d(0) > d^*$. We then obtain the desired behavior: (i) when $d \neq d^*$, the robot converges towards the spiral with $\alpha^*(t) = 0$ or $\alpha^*(t) = \pm\pi$ and (ii) when $d = d^*$, then $\alpha^*(t) = \alpha_c$ maintaining a constant distance.

Finally, it is necessary to compute the sign of α_c, *i.e.,* the direction of circumvention. We first compute the point cloud barycenter O_b. If its y coordinate is positive, then $\alpha_c = \pi/2$ and the circumvention direction is counter-clockwise. Otherwise $\alpha_c = -\pi/2$ and the circumvention direction is clockwise. Unlike the spiral center which is computed for each new laser scan, the circumvention direction is only calculated when the obstacle avoidance controller is started.

C. Obstacle avoidance controller

During the obstacle avoidance, the robot has to follow a succession of reference spirals while keeping the target in the camera field of view to restart the visual navigation once the obstacle is avoided. We present the two controllers achieving these tasks: the spiral following controller computes $\mathbf{q}_{sf} = [\upsilon, \omega_r]^T$ to make the mobile base follow a reference spiral and the pan-platform controller computes $\mathbf{q}_{pp} = \omega_{pp}$ to control the camera and keep the landmark in its field of view despite of the mobile base displacements.

To compute $\mathbf{q}_{sf}$, we first define the following error:

$$e_{sf}(t) = \alpha(t) - \alpha^*(t) \tag{9}$$

To make $e_{sf}(t)$ vanish, we impose an exponential decrease, that is: $\dot{e}_{sf}(t) = -\lambda_{sf} e_{sf}(t)$ where λ_{sf} is a positive scalar. We then derive an expression for $\dot{e}_{sf}(t)$ such as:

$$\begin{aligned} \dot{e}_{sf}(t) &= \dot{\alpha}(t) - \alpha_d \dot{\epsilon}(t) \\ &= -\omega_r(t) + \tfrac{\upsilon(t)}{d(t)} \sin(\alpha(t)) - \alpha_d \dot{\epsilon}(t) \end{aligned} \tag{10}$$

By imposing a constant linear velocity υ, and combining the previous equations, we obtain the following controller:

$$\omega_r(t) = \lambda_{sf}(\alpha(t) - \alpha^*(t)) + \frac{\upsilon}{d(t)} \sin(\alpha(t)) - \alpha_d \dot{\epsilon}(t) \tag{11}$$

Controller (11) only controls the robot mobile base. To keep the landmark in the camera field of view during the spiral following phase, we define the following new 1-dimensional error $e_{pp} = Y_g$ where Y_g is the abscissa of the visual pattern gravity center. Its time derivative is such as:

$$\dot{e}_{pp} = \mathbf{L}_g \nu_{\mathbf{c}} = \begin{bmatrix} \frac{f}{z_g} & \frac{Y_g}{z_g} & (f + \frac{Y_g^2}{f}) \end{bmatrix} \nu_{\mathbf{c}} \tag{12}$$

where z_g is depth of the Y_g. Partitioning $\mathbf{J}$ leads to: $\nu_{\mathbf{c}} = \mathbf{J}_{sf}\mathbf{q}_{sf} + \mathbf{J}_{pp}\mathbf{q}_{pp}$ where $\mathbf{J}_{sf}$ (respectively, $\mathbf{J}_{pp}$) is made of the two first columns (respectively, the third one) of $\mathbf{J}$. Imposing an exponential decrease such as $\dot{e}_{pp} = -\lambda_{pp} e_{pp}$, with $\lambda_{pp} > 0$, and replacing $\nu_{\mathbf{c}}$ by its expression in (12) allows to deduce:

$$\omega_{pp} = -\frac{1}{\mathbf{L}_g \mathbf{J}_{pp}} (\lambda_{pp} e_{pp} + \mathbf{L}_g \mathbf{J}_{sf} \mathbf{q}_{sf}) \tag{13}$$

Finally, the complete control vector for the obstacle avoidance phase is defined $\mathbf{q}_{oa} = \begin{bmatrix} \mathbf{q}_{sf}^T & \mathbf{q}_{pp} \end{bmatrix}^T$.

D. Finite-state machine

We now focus on the finite-state machine selecting the appropriate controller. It is made of 8 states and 11 transitions, and is shown in Fig. 4. Let us detail first the states, then the transitions. For the initial state S_0, the robot is stopped and ready to start the navigation. For S_1 and S_4, it moves towards the goal using the visual servoing controller, while for S_2 and S_3, it circumvents the hazard using the obstacle avoidance controller. Finally, for S_5, S_6 and S_7, the robot is stopped and the navigation is over. S_5 and S_6 are reached when the navigation fails due to respectively a collision or the target loss, while S_7 is reached when the navigation is a success.

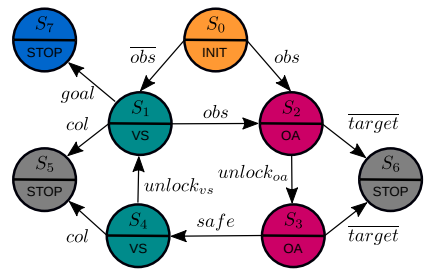


Fig. 4: Navigation finite-state machine

Regarding the transitions, obs is used to switch to the obstacle avoidance controller and is computed as follows:

$$obs = d < \delta_{d_{oa}} \text{ AND } |\alpha| < \delta_{\alpha_{oa}} \text{ AND } d > \hat{d} \tag{14}$$

where $\delta_{d_{oa}}$ and $\delta_{\alpha_{oa}}$ are respectively a distance and an angle defining a cone with its origin on O_l and centered on $\mathbf{x}_l$. Moreover, $\hat{d}$ is the predicted distance between O_s and O_r when $\mathbf{q}_{vf}$ is applied. Thus, the obstacle avoidance is activated when (i) O_s belongs to a user-defined cone, and (ii) the visual servoing drives the robot towards the obstacle. On the contrary, the nav condition is used to switch to the visual servoing controller and is computed as follows:

$$nav = |\theta_p| < \delta_{\theta_{vs}} \text{ AND } |\alpha| < \delta_{\alpha_{vs}} \tag{15}$$

where $\delta_{\theta_{vs}}$ and $\delta_{\alpha vs}$ are two angular thresholds. Thus, we switch to the visual servoing when (i) the pan-platform and the mobile base are almost aligned, *i.e.,* the robot is oriented towards the target, and (ii) O_s belongs to a user-defined cone.

To prevent too many switches between the controllers, the following strategy was developed. After switching from S_1 to S_2, it is not possible to immediately reactivate the visual servoing. The robot must first complete the spiral following transient phase. This is encoded in $unlock_{oa}$ as follows:

$$unlock_{oa} = |\alpha - \alpha^*| < \delta_{\alpha^*} \tag{16}$$

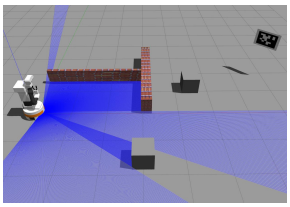
where δ_{α^*} is a user-defined threshold. Once $spiral$ is true, S_3 is reached, and it is possible to reactivate the visual servoing. Similarly, after switching from S_3 to S_4, it is not possible to immediately reactivate the obstacle avoidance. The robot must first move sufficiently away from the obstacle. This is encoded in $unlock_{vs}$ as follows:

$$unlock_{vs} = d > \delta_d \tag{17}$$

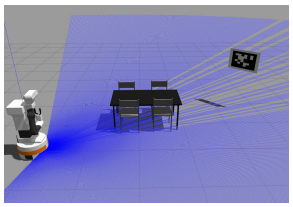
where δ_d is a user-defined threshold. Once $unlock_{vs}$ is true, S_1 is reached, and the obstacle avoidance can be reactivated.

The $target$, col and $goal$ conditions are used to stop the robot when necessary. They are respectively true when the landmark is in the camera field of view, a collision is detected, and the desired pose is reached.

IV. Results



(a) Environment A

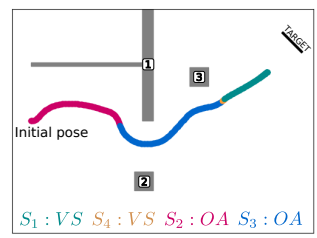


(b) Environment B

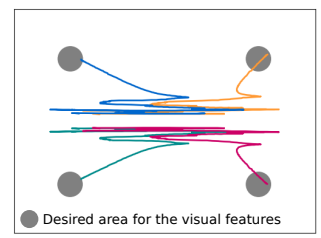
Fig. 5: The navigation environments

In this section we presents the results obtained when simulating the proposed navigation strategy with the ROS middleware, the 3D simulation software Gazebo, the Python programming language, and a robot model based on TIAGo from PAL Robotics. The robot is equipped with an RGB-D camera with a 1280 by 960 pixels resolution, and a laser rangefinder with a 220° field of view and a 0.33° resolution. The navigation strategy is tested in the two environments A and B (see Fig. 5). Environment A contains two convex cubes and a portion of wall forming a concave obstacle, and environment B is made of one table and four chairs. The obstacle of A are entirely detected by the laser rangefinder, while only the furniture legs can be sensed and appear as a set of small convex obstacles. For both environments, the robot has to reach a desired pose defined 1 meter in front of a landmark. This latter is made of an AprilTag [18] whose four corners are extracted using OpenCV to be used as the four visual features. Regarding the different parameters of the navigation the following values are used: $\Delta_o = 0.15m$ and $\Delta_m = 1m$ for the laser processing, $\lambda_{vf} = 0.5$, $\lambda_{sf} = 1.5$ and $\lambda_{pp} = 2$ for the control gains, $d^* = 0.6$ for the safety distance, and $\delta_{\alpha_{oa}} = 60°$, $\delta_{d_{oa}} = 5m$, $\delta_{\alpha_{vs}} = 80°$, $\delta_{\theta_{vs}} = 10°$ $\delta_{\alpha^*} = 10°$, and $\delta_d = 0.7m$ for the transition conditions. The navigation is stopped when the four visual features are at a distance smaller than 50 pixels from their reference values. This choice allows the robot to reach a pose sufficiently close to the desired one from a navigation perspective. To improve the robot pose, it would be necessary to switch to an IBVS with a gain higher than λ_{vs}. For the given configuration the control frequency is setup at 30 Hz.

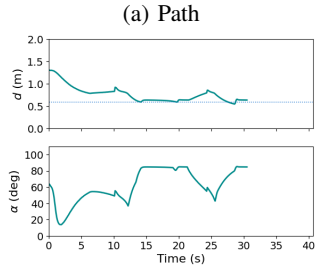
The results in environment A are shown in Fig. 6. In Fig. 6a, we can see that the first obstacle is immediately considered as dangerous, and the obstacle avoidance is started. The robot circumvents the obstacle without going deep into the concavity thanks to the calculation method of the spiral center proposed in this work. Once the robot gets closer to the second obstacle, the values of Δ_m and d^* allows passing between the two obstacles. Indeed, obstacles 1 and 2 are not merged based on Δ_m, and d^* is small enough to not make the robot collide with obstacle 2. On the contrary, obstacles 1 and 3 are merged based on Δ_m and the obstacle avoidance continues after passing the first obstacle. Once again, the robot does not deeply enter the concavity and manages to avoid the last obstacle. Finally, the visual servoing is started and the robot safely reaches an area around the desired pose. Indeed, in Fig. 6b, it can be seen that the task is successfully achieved in the image space. Moreover, by compensating the mobile base rotation with the pan-platform during the obstacle avoidance, the visual features are kept in the camera field of view during the entire navigation. In Fig. 6c, it can be seen that, while navigating in a complex environment, the robot successfully stays beyond the safety distance $d^* = 0.6m$, which is achieved by tracking a time-varying $\alpha^*(t)$. Finally, the control inputs leading to the described navigation are shown in Fig. 6d.



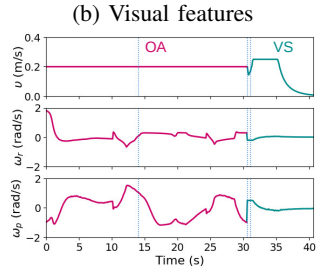
(a) Path



(b) Visual features



(c) Spiral parameters



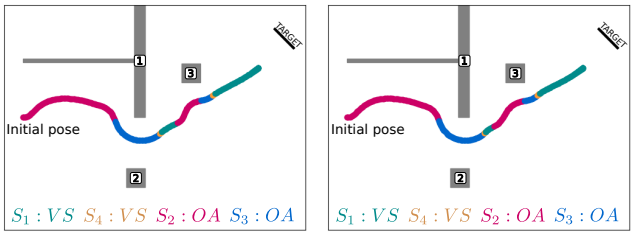
(d) Control inputs

Fig. 6: Results for environment A

The results obtained in environment A with two modifications of the navigation parameters are shown in Fig. 7. In Fig. 7a, the obstacle avoidance exit parameter is now equal to $\delta_{\alpha_{vs}} = 45°$. In Fig. 7b, the merging threshold is adjusted to $\Delta_m = 0.5m$, and obstacles 1 and 3 are not merged anymore. For both cases, it results in switching to the visual visual controller after passing the first obstacle. After a few moments, the third obstacle is considered as dangerous and the obstacle avoidance is re-started. Thus, the modification of two navigation parameters has some impact on the path but does not significantly modify the overall performances.

The results for environment B are shown in Fig. 8. Similarly to the previous case, the obstacle lies on the robot path towards the desired pose and the obstacle avoidance is immediately started (see Fig. 8a). However, from the laser rangefinder perspective the obstacle is not continuous but discrete, *i.e.,*

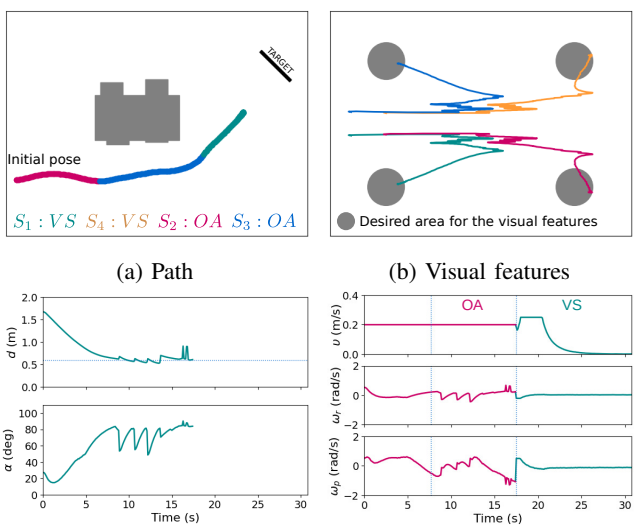
Video link



(a) Path with $\delta_{\alpha_{vs}} = 45°$ (b) Path with $\Delta_m = 0.5m$

Fig. 7: Results for environment A (cont.)

the furniture legs represent a set of small obstacles. Thanks to the merging step and the method used to compute the spiral center, the robot manages to circumvent the discrete obstacle with a behavior similar to the one obtained with a continuous obstacle. Once the obstacle is not dangerous anymore, the visual servoing is started to drive the robot towards a visual pose close to the desired one (see Fig. 8b). Similarly to the previous cases, it can be seen that the robot successfully stays beyond the safety distance d^* by tracking a time-varying $\alpha^*(t)$ (see Fig. 8c) and despite the discontinuities observed on both d and α. They are due to the limited laser field of view. Indeed, when the spiral center is at $\alpha = \pm\pi/2$, the obstacle is at the border of the laser field of view. Thus, when a furniture leg disappear from the laser view, the spiral center jumps to the next leg, creating discontinuities in the spiral center evolution. These discontinuities impact the control inputs (see Fig. 8d) but they do not cause the navigation to fail. This phenomenon can be avoided by increasing the sensor field of view.



(a) Path (b) Visual features

(c) Spiral parameters (d) Control inputs

Fig. 8: Results for environment B

V. Conclusion

In this paper we have presented a stand-alone obstacle avoidance method based on the follow-up of successive spirals. It allows dealing with unknown convex and concave obstacles while navigating in a cluttered environment. This method can be coupled with navigation controllers working in a variety of state spaces. We have shown an example where the navigation is performed using an IBVS controller and a finite-state machine is used to select the appropriate controller. The obtained results are satisfactory: the robot achieves the navigation while avoiding the unknown convex/concave obstacles without requiring any metric localization. In a near future, we plan to adapt the method to deal with small dynamic obstacles.

References

[1] J. Minguez, F. Lamiraux, and J.-P. Laumond, "Motion planning and obstacle avoidance," in *Springer handbook of robotics*. Springer, 2016, pp. 1177–1202.

[2] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.

[3] J. Borenstein, Y. Koren, *et al.*, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE transactions on robotics and automation*, vol. 7, no. 3, pp. 278–288, 1991.

[4] J. Minguez, "The obstacle restriction method (orm): Obstacle avoidance in difficult scenarios," in *IEEE Int. Conf. on Intelligent Robot and Systems*, 2005.

[5] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[6] D. Wilkie, J. Van Den Berg, and D. Manocha, "Generalized velocity obstacles," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 5573–5578.

[7] F. Chaumette and S. Hutchinson, "Visual servo control, part 1 : Basic approaches," *Robotics and Automation Mag.*, vol. 13, no. 4, 2006.

[8] A. Khelloufi, N. Achour, R. Passama, and A. Cherubini, "Tentacle-based moving obstacle avoidance for omnidirectional robots with visibility constraints," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1331–1336.

[9] F. Michaud and M. Nicolescu, "Behavior-based systems," in *Springer handbook of robotics*. Springer, 2016, pp. 307–328.

[10] K. N. Boyadzhiev, "Spirals and conchospirals in the flight of insects," *The college mathematics Journal*, vol. 30, no. 1, pp. 23–31, 1999.

[11] A. Mcfadyen, A. Durand-Petiteville, and L. Mejias, "Decision strategies for automated visual collision avoidance," in *International Conference on Unmanned Aircraft Systems*. IEEE, 2014, pp. 715–725.

[12] A. Durand-Petiteville, E. Le Flecher, V. Cadenat, T. Sentenac, and S. Vougioukas, "Design of a sensor-based controller performing u-turn to navigate in orchards," in *International Conference on Informatics in Control, Automation and Robotics*, vol. 2, 2017, pp. 172–181.

[13] A. Villemazet, A. Durand-Petiteville, and V. Cadenat, "Autonomous navigation strategy for orchards relying on sensor-based nonlinear model predictive control," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2022.

[14] D. Leca, V. Cadenat, T. Sentenac, A. Durand-Petiteville, F. Gouaisbaut, and E. Le Flecher, "Sensor-based obstacles avoidance using spiral controllers for an aircraft maintenance inspection robot," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 2083–2089.

[15] A. Durand-Petiteville, M. Courdesses, and V. Cadenat, "A new predictor/corrector pair to estimate the visual features depth during a vision-based navigation task in an unknown environment," in *7th International Conference on Informatics in Control, Automation and Robotics*, Funchal, Portugal, June 2010.

[16] Espiau, Chaumette, and Rives, "A new approach to visual servoing in robotics," *IEEE Trans. Robot. Automat.*, vol. 8, pp. 313–326, 1992.

[17] Pissard-Gibollet and Rives, "Applying visual servoing techniques to control a mobile handeye system," in *IEEE Int., Conf. on Robotics and Automation*, Nagoya, Japan, 1995.

[18] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4193–4198.