Sensor-based Obstacles Avoidance Using Spiral Controllers For an Aircraft Maintenance Inspection Robot

D. Leca[1,2], V. Cadenat[1,2], T. Sentenac[1,4], A. Durand-Petiteville[3], F. Gouaisbaut[1,2], E. Le Flécher[1,2]

***Abstract*— This paper deals with the problem of obstacle avoidance during an automated preflight inspection. During this mission, safety appears to be a key issue, as numerous obstacles are lying in a close neighborhood of the aircraft. They are very different in terms of size, shape and mobility and are often unforeseen. To cope with this highly evolutive environment, it is necessary to design a method sufficiently generic to deal with the obstacles variety and efficient enough to guarantee non collision and avoid classical problems such as local minima, singularities, etc. In this paper, we have proposed a new sensor-based control strategy fulfilling these two requirements. It consists in defining and following an adaptative spiral around the encountered obstacles while performing preflight inspection. It relies on two main elements: (i) the definition of a spiral whose parameters are continuously updated depending on the robot motion and on the environment; (ii) the coupling of two sensor-based controllers allowing to track the spiral while avoiding singularities. Experimental results conducted on our robot show the relevance and the efficiency of the proposed control strategy.**

I. Introduction

With nearly one hundred thousand commercial flights per day and one thousand aircraft inspections per hour across the globe, reducing the aircraft downtime and maintenance costs represent a challenging topic for the robotic community [1]. Because 70% of the inspections are still visually conducted by co-pilots and maintenance operators, the Air-Cobot project has been developed [1]. Its objective is to develop a mobile robot working in collaboration with agents to automate visual inspection procedures. This robot is intended to help co-pilots and maintenance operators to perform their duties faster, more reliably with repeatable precision. To realize visual inspections, the Air-cobot system has to autonomously navigate on the airport tarmac. This environment is a highly dynamical workplace: aircrafts are driving back and forth, human agents are working in their vicinity (refueling, freight loading/unloading, passengers boarding/disembarking, ...). Moreover, the encountered obstacles belong to different categories depending on their shape and size (aircrafts, trucks, freight, humans) and on their mobility: static (buildings), semi-static (freight that can be moved) or mobile (pedestrians, vehicles, etc.). Thus, the Air-Cobot robot must show strong autonomous capabilities by adapting its own behaviour to the encountered obstacles.

[1] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[2] Univ de Toulouse, UPS, LAAS, F-31400, Toulouse, France
[3] Department of Biological and Agricultural Engineering, University of California, Davis, CA, 95616, USA
[4] Institut Clément Ader (ICA), Université de Toulouse, CNRS, Mines Albi, UPS, INSA, ISAE-SUPAERO, Campus Jarlard, F-81013 Albi CT Cedex 09, France

To deal with this highly scalable and dynamical environment, the navigation strategy developed in the Air-Cobot project relies on both metric and topological maps. The first one provides a metric representation of the airport and is used to help the robot to localize itself, *i.e.,* to estimate its state in the Cartesian space. This latter is then used to calculate the controller allowing the robot to move from the hangar to the tarmac. Airports being made of roads and buildings which are well-known and not subject to any changes or modifications, metric maps are therefore well adapted to model them [2], [3], [4]. The topological map is used when the robot reaches the to-be-inspected aircraft neighborhood. Indeed, due to the presence of previously unforeseen obstacles, it is challenging to maintain an up-to-date metric map of the environment, although some improvements have been done [5], [6], [7]. For this reason, a topological map, made of the pre-flight inspection checkpoints, appears to be more suitable. It then offers a representation of the goals to reach, robust to the presence of previously unforeseen obstacles, and minimizes the need for updates of the scene model. Because of this minimal representation of the environment, it is usually suitable to couple these maps with sensor-based controllers to navigate. Because of their reactivity, these controllers easily deal with unknown environments, such as obstacles, and mostly rely on the acquired data to achieve their task.

In this paper, it is proposed to develop sensor-based controllers guaranteeing non-collision with unforeseen obstacles, when navigating in the aircraft vicinity between two checkpoints. Instead of using already developed obstacle avoidance techniques such as potential fields [8], [9], or Vector Field Histogram (VFH) [10], [11], [12], which are known to be sensitive to local minima, one proposes to investigate obstacle avoidance based on splines [13] or on spiral following. Indeed, relying on the spiral model presented in [14], obstacle avoidance techniques for unmanned aerial vehicles were developed in [15] and [16]. Moreover, spiral-based avoidance was also used for a ground robot in [17] and [18]. In these works, the choice for spiral following is mostly motivated by (i) the reactivity of the controller and (ii) the possibility of using the same method with several sensors. In this paper, our goal is to extend the works presented in [17], where the authors have defined the avoidance spiral using a unique center. This choice reduces the method generality which is restricted to the sole static obstacles with a geometric shape either circular or with a geometric aspect ratio close to one. Furthermore, the robot is controlled using a simple PID controller, reducing

performances. Thus, in order to improve both the method generality and the control performances, a solution relying on a suitable dynamical update of the spiral parameters is proposed. It consists in re-computing the spiral shape and center using both sensory data and the robot motion. In the proposed approach, the spiral center point (SCP) is moved with respect to the robot motion and to the newly-discovered parts of the obstacle, making the technique be able to handle any kind of obstacles. In addition to the SCP choice, it is also necessary to design a suitable sensor-based control law, to fulfill the following requirements: (i) production of a stable and smooth trajectory around the obstacles and (ii) generation of a control input compatible with the robot kinematics. The proposed control strategy extends previous works done in [17] and [18]. It relies on the switch between two sensor-based controllers: the first one, based on a state-to-input linearization, is presented for the first time in this paper, whereas the second one was already introduced in [18]. Thus, the proposed solution takes into account their respective advantages and drawbacks to overcome singularities and local minima issues. The performances are then significantly increased.

The paper is organized as follows. Sections II and III introduce the spiral model and the spiral parameters update, while sections IV and V are focused on the control design and the navigation strategy. Finally, section VI presents experimental results showing the interest of the approach.

II. Problem Modeling

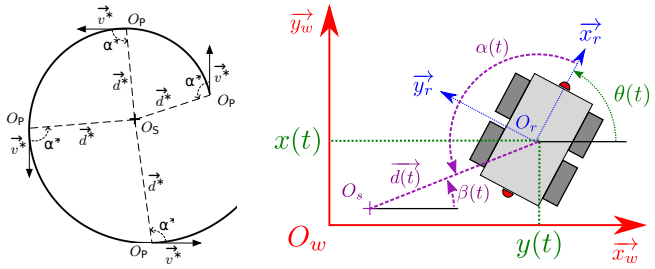
A. Spiral modeling

The section focuses on parameters used to define the equiangular spiral. As shown in figure 1a, the spiral is described by the point O_p moving on a plane with respect to a fixed point O_s. This point is considered as the center of the spiral. $\overrightarrow{v^*}$ is the velocity vector applied to O_p and its norm is denoted $v^*(t)$. Moreover $\overrightarrow{d^*}$ is the vector connecting O_s to O_p whose norm is $d^*(t)$. Finally $\alpha^*(t)$ is defined as the oriented angle between $\overrightarrow{v^*}$ and $\overrightarrow{d^*}$.

In [14] it is shown that if both $v^*(t)$ and $\alpha^*(t)$ are constant then O_p describes a spiral whose center is O_s. They are then respectively denoted by v^* and α^* in the sequel. Moreover [14] provides an important equation regarding $d^*(t)$:

$$\dot{d}^*(t) = -v^* \cos(\alpha^*) \tag{1}$$

Analyzing the above equation shows that the executed spiral only depends on the value of parameter α^*. Its sign allows to define the sense of motion (clockwise if negative, anticlockwise if positive), while its value fixes the type of spiral: inward if $0 \leq \alpha^* < \pi/2$ or $0 \leq \alpha^* < -\pi/2$, outward if $\pi/2 < \alpha^* \leq \pi$ or $-\pi/2 < \alpha^* \leq -\pi$, circle if $\alpha^* = \pm\pi/2$. From this analysis, it follows that this concept can be easily adapted to perform an obstacle avoidance motion. Indeed, fixing the spiral center point (SCP) on the obstacle surface and selecting a suitable couple of α^* and d^* allows us first to choose the sense of motion for the avoidance around the obstacle, and then to control the desired distance d^* and its evolution. This paper states how to choose these parameters from the available sensory data to perform an efficient and safe avoidance motion.



(a) Spiral model (b) Robot model

Fig. 1: Spiral and robot modelling

B. Robot modelling

The Air-Cobot robot is based on the Sterela 4MOB platform. It has four-wheel skid steering drive, which allows the robot to turn on itself. Its model is presented in figure 1b. $F_w = (O_w, \overrightarrow{x_w}, \overrightarrow{y_w}, \overrightarrow{z_w})$ is the frame linked to the world, while $F_r = (O_r, \overrightarrow{x_r}, \overrightarrow{y_r}, \overrightarrow{z_r})$ is the frame attached to the robot. $\chi(t) = [x(t), y(t), \theta(t)]^T$ represents the pose of the robot in the world frame, where $x(t)$ and $y(t)$ are the coordinates of O_r in F_w and $\theta(t)$ is the angle between $\overrightarrow{x_w}$ and $\overrightarrow{x_r}$. O_s is the center of the spiral to be followed, while $\overrightarrow{d}$ represents the vector connecting O_s to O_r and $\alpha(t)$ is the angle between $\overrightarrow{x_r}$ and $\overrightarrow{d}$. $\beta(t)$ is the angle between $\overrightarrow{x_w}$ and $\overrightarrow{d}$. It should be noticed that:

$$\alpha(t) = \pi - \theta(t) + \beta(t) \tag{2}$$

$$\dot{\alpha}(t) = -\dot{\theta}(t) + \dot{\beta}(t) = -\omega(t) + \frac{v(t)}{d(t)} \sin(\alpha(t)) \tag{3}$$

Consequently, $d(t)$ represents the distance between O_s and O_r. It is shown in [14] that:

$$\dot{d}(t) = -v(t) \cos(\alpha(t)) \tag{4}$$

III. Spiral Parameters Update

A. Choice of α^*

The choice of α^* allows to define the way the robot will move with respect to the obstacle. An angle lower than $\frac{\pi}{2}$ (inward spiral) makes the robot go closer and closer to the obstacle, leading to potential collision. An angle greater than $\frac{\pi}{2}$ (outward spiral) is not suitable either, because it makes the robot go further and further from the obstacle, leading to a longer trajectory. Therefore, we have imposed $\alpha^* = \pm\frac{\pi}{2}$. The sign of α^* depends on the sense of motion around the obstacle, and will be discussed later in section V.

B. Choice of d^*

d^* has been fixed to a constant value to be consistent with the choice of α^*, according to equation (4). It is chosen to ensure that the robot will not move closer than this distance to any detected obstacle.

C. Choice of the SCP

We propose to define a moving SCP instead of a static one as in [1] and [18]. The SCP must be chosen to guarantee that: (i) the non collision and the robot safety are insured ; (ii) obstacles of various shapes (convex or concave, with sharp angles, etc.) and sizes (small or large) are handled in an equal way ; (iii) a smooth trajectory of the robot around the obstacles with important variations of shape is guaranteed, which means that the potential unknown parts of the obstacle have to be taken into account ; (iv) the control input must suit the robot kinematics. The SCP evolution thus depends on the robot motion and on the LiDAR data.

a) Algorithm: The proposed method consists in computing the spiral center point using these three steps :
- Step 1 : After a LiDAR acquisition around the robot, the closest point from the robot O_c is computed (see figure 2).
- Step 2 : All the LiDAR points within a $2d^*$ meters radius circle centered on O_c are used to compute their barycenter O_b. This $2d^*$ radius circle is taken in order to encompass the safety distance around the robot (blue circles on figure 2).
- Step 3 : The distance d_b (respectively, d_c) between O_b (respectively, O_c) and the robot is computed. The SCP O_s is given by the closest point to the robot, that is: O_b if $d_b < d_c$ and O_s if $d_b \geq d_c$.

b) SCP motion for different obstacles: The figure 2 shows how the SCP is computed for different kinds of obstacles during the avoidance. In green are displayed the LiDAR points, while the blue and red circles respectively materialize the robot safety distance and the $2d^*$ meters radius circle centered on O_c which is used to compute the barycenter. Concerning case (a.), the algorithm first computes O_c which is the orthogonal projection of the robot on the wall. Second, O_b is computed as the barycenter of all the points which are within the red circle. In this case, because only one segment is perceived, O_b coincides with O_c, which leads to: $O_s = O_b = O_c$. In case (b.), O_c is at the corner of a squared (thus convex) obstacle. The barycenter O_b then lies inside the obstacle shape. Consequently, $O_s = O_c$. Concerning case (c.), similarly to (a.), O_c is obtained on the wall. The green points lying in the red circles are then considered to deduce the barycenter and are used to compute point O_b. This latter being closer to the robot than O_c, it follows that $O_s = O_b$. Thus, in case of straight or convex obstacles, $O_s = O_c$ and O_s follows the edges of the obstacle. As for concave obstacles, $O_s = O_b$ and O_s is in the free space. This nice property anticipates the forthcoming concave shape of the obstacle, allowing to prevent local minima.

IV. Design of the avoidance control laws

To perform the avoidance, the robot must first reach the desired spiral, and then follow it around the obstacle. Two controllers are proposed. The first one is designed using an exact input to state linearization method. The second one is inspired by [18]. As it will be highlighted, the first controller is efficient at maintaining the robot on the correct trajectory, but suffers from singularities. The second one is singularity-free, and is more suitable during the approach phase.

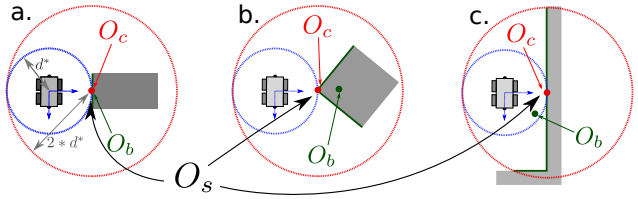


Fig. 2: SCP computation for three kinds of obstacles: (a.) Straight. - (b.) Convex. - (c.) Concave.

A. Definition of the errors

To properly avoid obstacles, it is necessary to keep the robot at a given distance and orientation from them. To do so, the two following errors must be vanished: $e_\alpha = \alpha(t) - \alpha^*$ and $e_d = d(t) - d^*(t)$. In [18], in order to track a spiral defined by its SCP together with v^* and α^*, it is proposed to impose $v(t) = v^* \neq 0$.

B. First controller design

This section proposes to design an output feedback control law which makes the errors e_α and e_d asymptotically converge toward zero. As the system is nonlinear, the main idea is to use an exact input-to-state linearization method proposed by [19]. The errors dynamics are defined by:

$$\begin{cases} \dot{e}_d(t) = v^\star \left[\cos(\alpha^\star) - \cos(\alpha(t))\right] \\ \dot{e}_\alpha(t) = -\omega(t) + \frac{v^\star}{d(t)} \sin(\alpha(t)) \end{cases} \quad (5)$$

Which can be written as follows:

$$\begin{cases} \dot{e}_d(t) = v^\star \left[\cos(\alpha^\star) - \cos(e_\alpha(t) + \alpha^\star)\right] \\ \dot{e}_\alpha(t) = -\omega(t) + \frac{v^\star}{e_d(t) + d^\star(t)} \sin(e_\alpha(t) + \alpha^\star) \end{cases} \quad (6)$$

The main idea is to linearize the error system. To this end, consider the transformation:

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} e_d(t) \\ v^\star \left[\cos(\alpha^\star) - \cos(e_\alpha(t) + \alpha^\star)\right] \end{bmatrix} = T(e) \quad (7)$$

Notice that $T(0) = 0$ and in the domain $\mathcal{D}$ defined by:

$$\mathcal{D} = \left\{(e_d, e_\alpha) \in \Re^2 \| e_d \in \Re, e_\alpha \in]-\alpha^\star, -\alpha^\star + \pi[\right\} \quad (8)$$

T defines a diffeomorphism. Applying this transformation to the error system leads to:

$$\begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = v^\star \sin(e_\alpha(t) + \alpha^\star)(-\omega(t) + \frac{v^\star \sin(e_\alpha(t) + \alpha^\star)}{e_d(t) + d^\star(t)}) \end{cases}$$

Taking

$$\widetilde{\omega}(t) = v^\star \sin(e_\alpha(t) + \alpha^\star)(-\omega(t) + \frac{v^\star \sin(e_\alpha(t) + \alpha^\star)}{e_d(t) + d^\star(t)})$$

we obtain :

$$\begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = \widetilde{\omega}(t) \end{cases} \quad (9)$$

where $\widetilde{\omega}(t)$ is a new control law, which has to be designed. At this stage, as the system (9) is linear, a classical linear control law:

$$\widetilde{\omega}(t) = -\lambda_1 \; z_1(t) - \lambda_2 \; z_2(t) \tag{10}$$

with $\lambda_1, \lambda_2 > 0$, allows to stabilize asymptotically system (9). Finally, from (9) and (10), the following control law ω is obtained:

$$\omega(t) = -\frac{\widetilde{\omega}(t)}{v^\star \sin(e_\alpha(t) + \alpha^\star)} + \frac{v^\star \sin(e_\alpha(t) + \alpha^\star)}{e_d(t) + d^\star(t)} \tag{11}$$

In the sequel, this control law will be denoted by ω_A. Now, we propose the following theorem:

Theorem 1: Consider two positive scalars, λ_1, λ_2, the error system (5) in closed loop with the control law (9), (9), (10), where $z = T(e)$ in (7) is locally asymptotically stable.

Proof: It is sufficient to notice that z converges asymptotically to zero and $z = T(e)$ defines a local diffeomorphism with $T(0) = 0$. ■

Remark 1: By construction, this controller will maintain the robot along the desired spiral. λ_1 and λ_2 are used to tune the speed of convergence of the closed loop system

Remark 2: The term $\sin(e_\alpha(t)+\alpha^*)$ at the denominator of equation (11) introduces singularities when $\alpha(t) = k\pi, k \in \mathbb{N}$. This term appears because of the second order derivation of the error $e_d(t)$. Thus, the idea is to define a new error to obtain a singularity-free controller.

C. Second controller design

The previous analysis has highlighted the need for a another controller. It has been shown that the singularities in ω_A mainly come from the use of the second derivative of $e_d(t)$ in the controller design. So, the first idea would be to define an error independent of this term. However, in this case, the robot reaches a spiral which is not necessarily the one located at the right distance from the obstacle, as shown in [18]. Thus, this solution is not suitable because the so-designed controller does not allow to control the distance error $e_d(t)$. It is then necessary to keep a term depending on the distance in the error to ensure a safe distance from the obstacle. We introduce the following hybrid error [18]:

$$e_S(t) = e_\alpha(t) - \alpha_S(t, d) \tag{12}$$

$e_S(t)$ relies on $e_\alpha(t)$, and on a term $\alpha_S(t, d)$ depending on time t and on the current distance $d(t)$ between the robot and the obstacle. Now that the error to be vanished has been defined, we focus on the controller design problem. As a first step, we propose to compute the error derivative with respect to time to identify the terms involved in its dynamics:

$$\dot{e}_S(t) = -\omega(t) + \frac{v(t)}{d(t)} \sin(\alpha(t)) - \dot{\alpha}_S(t, d) \tag{13}$$

In order to make $e_S(t)$ vanish, we impose an exponential decrease. It leads to the following controller ω_B:

$$\omega_B(t) = \lambda_S e_S(t) + \frac{v(t)}{d(t)} \sin(\alpha(t)) - \dot{\alpha}_S(t, d) \tag{14}$$

where λ_S is a positive scalar. As we can see, the control input is singularity-free if the term $\dot{\alpha}_S(t, d)$ is singularity-free too. We propose the following definition for $\alpha_S(t, d)$:

$$\alpha_S(t, d) = \epsilon(t)\alpha_D \tag{15}$$

$\epsilon(t)$ is the normalized error between $d^*(t)$ and $d(t)$. It has been saturated to ± 1:

$$\epsilon(t) = \text{sign}(d^*(t) - d(t)) \frac{\min(|d^*(t) - d(t)|, n)}{n} \tag{16}$$

where $n \in \mathbb{N}^*$. $\epsilon(t)$ belongs to the domain $[0, 1]$ if $d^*(t) > d(t)$ or $[-1, 0]$ if $d^*(t) < d(t)$. Indeed, $\epsilon(t)$ has its norm equal to 1 when $||d(t) - d^*(t)||$ is greater than an arbitrary value n, and equal to 0 when $d(t) = d^*(t)$. Additionally, α_D is defined as:

$$\alpha_D = \begin{cases} \text{sign}(\alpha^*) \times \pi - \alpha^* & \text{if } d^*(0) > d(0) \\ \alpha^* & \text{if } d^*(0) < d(0) \end{cases} \tag{17}$$

As shown in [18], this controller is guaranteed to be locally asymptotically stable once the $\alpha(t)$ overpasses $\alpha^*(t)$.

D. Switching strategy

Two controllers $\omega_A(t)$ and $\omega_B(t)$ have been previously introduced. The $\omega_A(t)$ controller suffers from a singularity problem which occurs when α vanishes. This problem might happen in two cases: (i) During the start of the avoidance process, if the robot is facing the obstacle ; (ii) During the avoidance motion, if a sudden variation of the environment (due to a corner for example) occurs, leading to a sudden variation of the SCP which in turn brings α close to 0. The following switching conditions have been chosen to insure a proper use of $\omega_A(t)$.

1) Switching conditions: At the beginning of the obstacle avoidance, the robot uses the second controller $\omega_B(t)$ to avoid any singularity problem (see the above case (i)). As previously mentioned, this controller makes the vehicle converge towards the spiral. When the error $e_\alpha(t)$ drops below a threshold e_α^{switch}, *i.e.*, when it is guaranteed that the value of this angle is far enough from the singularity, the $\omega_A(t)$ controller is applied to the robot. This reasoning leads to the following switching conditions:

- If $|e_\alpha(t)| < e_\alpha^{switch}$, use the ω_A controller.
- Else use the ω_B controller.

Note that e_α^{switch} must be set low enough to keep $\omega_A(t)$ within admissible limits. In addition, it allows to express how far from the singularity α must be before applying $\omega_A(t)$ controller. Finally, some chattering may occur between ω_A and ω_B when $e_\alpha(t) \simeq e_\alpha^{switch}$, because of the small variations of $\alpha(t)$ due to the sensor noise and the environment evolution. To prevent this phenomenon, the switching threshold is implemented as an hysteresis switching function.

2) Control law smoothening: When a switch occurs, a discontinuity in the control inputs may appear. To deal with this problem, a moving average window has been implemented to smoothen the resulting control law. Thus, when the switch between ω_A and ω_B is activated, the

resulting angular control ω_R is computed as follows:

$$\omega_R(t) = \frac{i}{p} \times \omega_A^{last} + \frac{p-i}{p} \times \omega_B(t) \quad (18)$$

ω_A^{last} is the last ω_A command computed before the switch, $p \in \mathbb{N}^*$ is the number of iterations used to handle the switch and $i \in [0, ..., p]$.

V. Navigation strategy

As explained before, the navigation task consists in reaching a goal O_g whose coordinates are defined by (x_g, y_g) in the world frame F_w. To do so, two controllers are applied to the robot: a Go-To-Goal controller (GTG) in the free space and the above mentioned Spiral Avoidance controller (SA) when a collision risk occurs. The first one is a basic proportional controller correcting the heading of the robot to reach O_g (the orientation problem is not considered here). A supervisor is responsible for switching between these two control laws depending on the LiDAR data. Considering three angles α_b, α_c and α_g such as:

$$\alpha_b = (\overrightarrow{x_r}, \overrightarrow{O_rO_b}) \qquad \alpha_c = (\overrightarrow{x_r}, \overrightarrow{O_rO_c})$$
$$\alpha_g = (\overrightarrow{x_r}, \overrightarrow{O_rO_g}) = \text{atan2}(y_g - y, x_g - x) - \theta$$

We define two guarding conditions (1) and (2) :

$$(1) \begin{cases} d_c < d_c^{tr} \text{ and} \\ \mid \alpha_g - \alpha_c \mid < \pi/2 \end{cases} \qquad (2) \begin{cases} d_b < d_b^{tr} \text{ and} \\ \mid \alpha_g - \alpha_b \mid < \pi/2 \end{cases}$$

with d_c^{tr}, d_b^{tr} two trigger distances defined such as :

$$\left.\begin{array}{l} d_c^{tr} = d^* + d^*(1 - \frac{|\alpha_c|}{\pi/2}) \\ d_b^{tr} = d^* + d^*(1 - \frac{|\alpha_b|}{\pi/2}) \end{array}\right\} if\ GtG\ controller$$
$$\left.\begin{array}{l} d_c^{tr} = 2d^* \\ d_b^{tr} = 2d^* \end{array}\right\} if\ SA\ controller$$

The SA controller is active if at least one of the conditions (1) or (2) is true. Two cases need to be separated :

- The robot is using the GtG controller : d_c^{tr} and d_b^{tr} are between d^* and $2d^*$. These values depend on the angles with which the robot enters the obstacle neighborhood. If the obstacle is in front of the robot, the avoidance should be triggered early (at $2d*$). If it is not the case, there is no need for such an anticipation.
- The robot is using the SA controller : d_c^{tr} and d_b^{tr} are both equal to $2d^*$.

In order to avoid any discontinuity during the switch, a moving average window technique similar to the one presented in section IV has been implemented. When this switch occurs, the avoidance sense of motion (SOM) is determined. This sense of motion is chosen to lead to the shortest avoidance trajectory around the obstacle, based on the knowledge of the obstacle the robot has when the avoidance is activated. Consequently, the values of α_b and α_g are compared to choose the adequate sense of motion :

- If $\alpha_b \leq \alpha_g$, the largest part of the obstacle lies on the robot left hand side. Avoidance must then be performed following the obstacle right hand side and the SOM is counter-clockwise.
- If $\alpha_b > \alpha_g$, the largest part of the obstacle lies on the robot right hand side. The SOM is then clockwise.

Remark 3: The SOM is chosen when the switch to the SA controller occurs, and remains the same until the GTG controller is applied again. Since the robot operates at a constant linear velocity $v(t) = v^* \neq 0$, it will keep going around the obstacles until the leaving condition becomes true. Thus, it is not possible for the robot to become stuck in a local minima.

Figure 3 exhibits two examples :

- Example (a) : the obstacle is small and spherical, thus $O_b \simeq O_c$. In this case $\alpha_b < \alpha_g$ and a counter clockwise sense of motion is chosen.
- Example (b) : several obstacles lie within the LiDAR sensor range. Because these obstacles are closer from $2d^*$ to each others, they are all taken into account to compute O_b. Consequently, $\alpha_b < \alpha_g$, and a clockwise sense of motion is chosen.

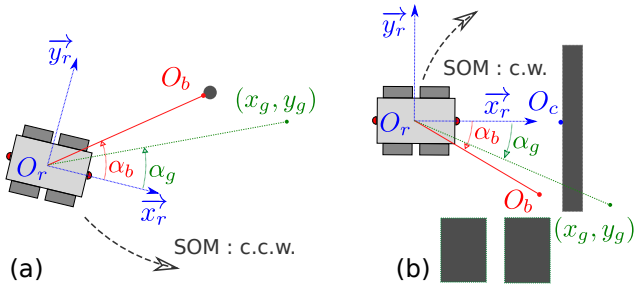


Fig. 3: Choice of the sense of motion

VI. Experiments and results

The section describes the experimentation of the navigation of Air-Cobot in an environment with different obstacles (small obstacles compared to people, car and building). The performances of the robot in terms of trajectory, errors and control are analysed.

A. Robot presentation

The robot is equipped with an embedded unit (carrying the computer), and several sensors. Among these sensors, three of them will be used for this experiment: an Inertial Measurement Unit (IMU), four incremental sensors on the wheels (odometry) and two Hokuyo UTM-30LX Laser Range Finders (LRF), providing a 360° information about the robot environment, at a 0.25° angular resolution. Using an Extended Kalman Filter, the informations from the IMU and the odometry are merged to compute a resulting pose information. This pose information is given with respect to the initial frame of the robot.

B. Experiment of navigation in an cluttered environment

This experiment aims to show the robot ability to move in an airport-like environment, by detecting and avoiding obstacles. Experimental tests on a real airport environment

are costly and require a lot of logistic. Thus, an environment has been designed at LAAS-CNRS in order to reproduce situations the robot could be facing in an airport area. As shown in figure 4, starting from its initial position O_w, the robot has to move toward a specified goal O_g whose coordinates in the initial robot frame are $[x_g, y_g] = [60, -5]$. Between the robot and its goal lay several obstacles : a cluster of small obstacles (trash bins, lamp, tank, etc...) (A), a parked car (B) and two big buildings (C) and (D) (see figure 4). This outdoor environment represents the typical area the robot will need to navigate through.

The linear velocity v^* has been fixed to $0.25\ m.s^{-1}$. To ensure a safe motion taking into account the size of the robot (roughly 150x80cm), $d^* = 3m$ is chosen. The gains for the controllers are set as follows: $\lambda_1 = 0.1$, $\lambda_2 = 0.1$, $\lambda_S = 0.5$ and $n = 5$. The switch threshold e_α^{switch} is selected as $e_\alpha^{switch} = \frac{\pi}{12}$.

C. Analysis of the Air-Cobot performances

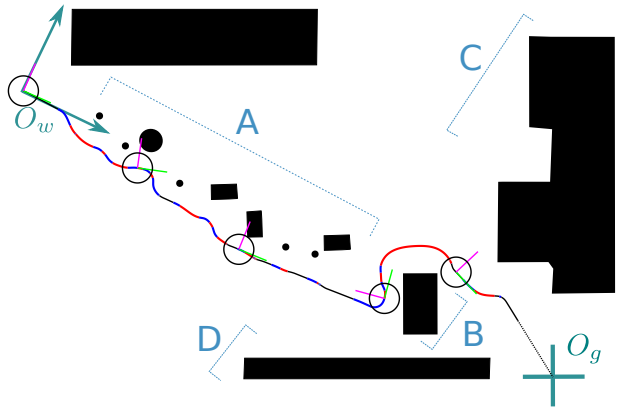


Fig. 4: **Robot trajectory during the real experiment**: in black line: GtG controller, in red line: $\omega_B(t)$ controller, in blue line: $\omega_A(t)$ controller.

The performances of the Air-Cobot are analysed through its trajectory, the errors, in particular the error in distance between the robot and the obstacle, and the control law resulting from different controllers. Figure 4 shows the trajectory of the robot in its environment. It can be seen that the robot is able to reach its goal while avoiding all obstacles: the car, the point obstacles and the building[1]. Figure 5-a. and 5-b. exhibits the evolution of the distance error $e_d(t)$ and of the angular error $e_\alpha(t)$. Figure 5-c. shows the evolution of the angular velocity as well as the selected current controller. The performances analysis is carried out according to different kind of obstacles as follows:

1) Avoidance of cluster of obstacles (A): As shown by figure 4, at the beginning, the robot starts at O_w. The area in front of it being all clear, it immediately heads toward the goal O_g using the GtG controller. At $t = 16s$, the avoidance is triggered. Using the conditions explained in section V, a counter-clockwise SOM is selected, because most of the visible obstacles is on the robot left hand side. Initially, there is a distance error $e_d(t) \simeq 1.5m$, due to the trigger conditions. Because the robot is nearly facing the first obstacle of the cluster (A), there is also an angular error $e_\alpha(t) \simeq -1.2rad$. This angular error is greater than $e_\alpha^{switch} = \pi/12 \simeq 0.26rad$. Consequently, according to condition IV-D.1, the second controller ω_B is used to start the avoidance. At $t \simeq 24s$, the angular error starts to be low enough to switch to the first controller ω_A. The switch occurs without any visible discontinuity on the control law thanks to equation (18). From $t = 16s$ to $t = 150s$, the robot avoids the succession of small obstacles that are in the cluster (A). The robot then switches several times between the two avoidance controllers in order to bypass each small obstacles. The cluster (A) being a group of close small obstacles, it can be seen on figure 4 that, thanks to the choice of the SCP, the robot is able to avoid this cluster as a whole: it does not try to enter the gap between each obstacle, thus avoiding the risk of getting stuck in a local minima. Because the environment is unforeseen and the SCP is evolving with it, the errors $e_d(t)$ and $e_\alpha(t)$ are variating around 0. At $t \simeq 150s$, the robot considers that the cluster (A) is avoided, and goes toward the goal using the GtG controller.

2) Avoidance of obstacle (B): At $t \simeq 164s$, the robot encounters the car (B). It decides a clockwise SOM, and starts using the ω_B controller before switching to the ω_A controller. The robot could have avoided (B) counter-clockwise through the gap between (B) and (D), but because this gap was smaller than $2d^*$, safety was not guaranteed. This highlights the interest of using O_g to determine the SCP. Because the distance between (B) and (D) was small enough, a part of (D) was taken for the computation of O_g, moving it to the robot left hand side, and leading to a clockwise SOM. From $t \simeq 192s$ and $t \simeq 225s$, ω_A controller is used and is able to bring and keep the errors close to 0. At $t \simeq 227s$, the car is avoided, and the robot goes back to the GtG controller until it encounters the building (C) at $t \simeq 235s$.

3) Avoidance of obstacle (C) and end of the experiment: At this time, the robot turns around the corners. Finally, it switches to the GtG controller and moves toward the goal.

This experiment validates several desired behaviors:

- *Safety:* As shown by figure 5, the distance error $e_d(t)$ has never dropped below -1.5m, which means that Air-Cobot has not gone closer than 1.5 meters from any obstacle. In spite of the unknown and highly-variable environment, the robot has succeeded in maintaining a safe distance to each obstacle. Additionally, the avoidance is triggered at a distance greater than d^*, in order to have enough time to turn before reaching $d = d^*$.
- *Low sensitivity to the obstacles size or shape:* Air-Cobot was able to avoid successfully a large cluster of obstacles, a car, and a large building. The robot can bypass inner and outer corners, cluster of several obstacles, as well as large walls. It is then able to deal with different kinds of objects.
- *Relevant navigation strategy:* The SOM is determined

[1]The full video of the experiment can be found at the following address: http://homepages.laas.fr/cadenat/ECC19/ECC19AircobotExperiment.mp4

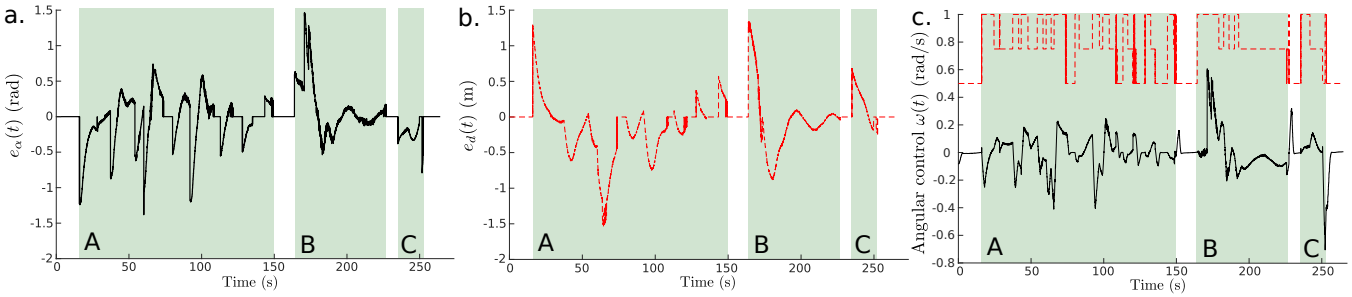


Fig. 5: a. Angular error (in rad) – b. Distance error (in meters) – c. Angular velocity and controller used. For the controller plot : GtG controller = 0.5; ω_B controller = 1 and ω_A controller = 0.75.

to produce the shortest path ensuring safety, taking into account the knowledge the robot has from the environment at the decision instant. Additionally, the robot does not extend the avoidance further than necessary. As soon as the robot considers the front area as free, it resumes to a GtG controller. Finally, by choosing a new SOM each time the SA controller is activated, and keeping it constant during the avoidance, it prevents the robot from being stuck in a local minima.

VII. Conclusion

This paper was focused on the problem of obstacle avoidance. It has proposed a novel control strategy sufficiently generic to deal with a large variety of obstacles and efficient enough to guarantee non collision and avoid classical problems such as local minima, singularities, etc. This strategy extends previous works and consists in defining and following a suitable spiral around the encountered obstacles during the mission. Two main contributions are at the core of this paper: (i) the continuous update of the spiral parameters by defining a suitable barycenter able to guarantee smooth trajectory and control inputs; and (ii) the coupling of two sensor-based controllers allowing to track the spiral while avoiding singularities. A complete navigation strategy has also been proposed by switching between the spiral based avoidance technique and the dedicated go to goal controller thanks to adequate guarding conditions. The approach has been experimented in LAAS on our Air-Cobot system. The obtained results have shown its relevance and interest for the considered mission.

However, the proposed approach is still limited to unforeseen static or semi-static objects. Therefore, our next step is to extend it to dynamic obstacles.

References

[1] M. Futterlieb, "Vision based navigation in a dynamic environment," Ph.D. dissertation, Universite de Toulouse, 2017.

[2] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion*. MIT Press, Boston, 2005.

[3] F. Bonin-Font, F. Ortiz, and G. Oliver, "Visual navigation for mobile robots : a survey," *Journal of intelligent and robotic systems*, vol. 53, no. 3, p. 263, 2008.

[4] C. Stachniss and W. Burgard, "An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02)*, Lausanne, Switzerland, October 2002.

[5] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *Robotics, IEEE Transactions on*, vol. 21, no. 3, pp. 354 – 363, june 2005.

[6] M. Khatib, H. Jaouni, R. Chatila, and J. P. Laumond, "Dynamic path modification for car-like nonholonomic mobile robots," in *Proceedings of International Conference on Robotics and Automation*, vol. 4, Apr 1997, pp. 2920–2925 vol.4.

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

[8] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2, Mar 1985, pp. 500–505.

[9] M. Khatib and R. Chatila, "An extended potential field approach for mobile robot sensor-based motions," in *Intelligent Autonomous Systems (IAS)*, I. Press, Ed., Karlsruhe, Germany, 1995, pp. 490–496.

[10] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 3, pp. 278–288, 1991.

[11] I. Ulrich and J. Borenstein, "Vfh+: Reliable obstacle avoidance for fast mobile robots," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 2. IEEE, 1998, pp. 1572–1577.

[12] ——, "Vfh*: Local obstacle avoidance with look-ahead verification," in *IEEE International Conference on Robotics and Automation*, San Francisco, May 2000.

[13] T. Mercy, R. Van Parys, and G. Pipeleers, "Spline-based motion planning for autonomous guided vehicles in a dynamic environment," *IEEE Transactions on Control Systems Technology*, vol. PP, pp. 1–8, 08 2017.

[14] K. N. Boyadzhiev, "Spirals and conchospirals in the flight of insects," *The College Mathematics Journal*, vol. 30, no. 1, pp. pp. 23–31, 1999. [Online]. Available: http://www.jstor.org/stable/2687199

[15] A. Mcfadyen, L. Mejias, and P. Corke, "Visual servoing approach to collision avoidance for aircraft," in *28th Congress of the International Council of the Aeronautical Sciences 2012*, Brisbane Convention & Exhibition Centre, Brisbane, QLD, September 2012. [Online]. Available: http://eprints.qut.edu.au/54328/

[16] A. Mcfadyen, P. Corke, and L. Mejias, "Rotorcraft collision avoidance using spherical image-based visual servoing and single point features," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct 2012, pp. 1199–1205.

[17] M. Futterlieb, V. Cadenat, and T. Sentenac, "A navigational framework combining visual servoing and spiral obstacle avoidance techniques," in *Informatics in Control, Automation and Robotics (ICINCO), 2014 11th International Conference on*, vol. 02, Sept 2014, pp. 57–64.

[18] E. L. Flecher, A. Durand-Petiteville, 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*, Madrid, Spain, July 2017.

[19] A. Isidori, *Nonlinear control systems*. Springer Science & Business Media, 2013.