DESIGN OF AN INTELLIGENT NAVIGATION STRATEGY TO DEAL WITH UNEXPECTED DYNAMIC OBSTACLES

A Renak, D Leca, Viviane Cadenat, Carine Jauberthie, A Durand-Petiteville

DESIGN OF AN INTELLIGENT NAVIGATION STRATEGY TO DEAL WITH UNEXPECTED DYNAMIC OBSTACLES

A. Renak*, D. Leca†, V. Cadenat†, C. Jauberthie†, A. Durand-Petiteville‡

* *Univ de Toulouse, UPS, F-31400, Toulouse, France*

† *Univ de Toulouse, UPS, LAAS, F-31400, Toulouse, France*
CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France

‡ *Universidade Federal de Pernambuco, Departamento de Engenharia de Mecânica,*
Av. Prof. Moraes Rego, 1235 - Cidade Universitária, Recife, PE, 50670-901, Brazil

Emails: mameziane.renak@gmail.com, dleca@laas.fr, cadenat@laas.fr, cjaubert@laas.fr, ajsd@cin.ufpe.fr

Abstract— This paper deals with the problem of navigating through a poorly known environment cluttered with both static and dynamic obstacles. The proposed strategy relies on two controllers allowing to reach the goal and to avoid the obstacles. Two contributions can be highlighted: (i) the definition of a safe avoidance distance which can be adequately modified during the mission if a moving obstacle is encountered; and (ii) the choice of a sense of motion depending on the obstacle motion. Simulation results validate the proposed strategy.

Keywords— Mobile robotics, Obstacle avoidance, Navigation strategy, Decision making, Adaptive threshold.

1 Introduction

The problem of making a mobile robot autonomously navigate has been the focus of numerous researches over the last decades. The proposed approaches can be divided into two categories: the map- based ones and the reactive ones (Choset et al., 2005). The first set of methods relies on a metric representation of the environment which is then used to plan a path/trajectory to follow. This navigation strategy is widely used and allows to perform long range navigation, i.e. when the goal to reach cannot be seen from the initial robot pose. However, when considering a dynamic navigation environment, it becomes more challenging to rely on these methods. Indeed, the model of the scene has to be updated while navigating, in order to include the dynamic elements, and the path to follow has to be re-computed. For these reasons, such approaches seem to reach certain limits when considering a dynamic environment.

The second set of methods relies on reactive controllers, which drive a robot based on local knowledge of the environment obtained by the embedded sensors. Such controllers offer suitable performances when dealing with unknown environment. For this reason, navigation frameworks based on reactive controllers have been designed to deal with challenging navigation environments (Segvic et al., 2007; Durand-Petiteville et al., 2015). Among the issues to overcome, one concerns the avoidance of unforeseen obstacles present in the scene. One of the common solution consists of coupling the navigation controller to a reactive controller performing obstacle avoidance. Thus, depending on the robot location relative to the obstacles, one uses either the navigation controller or the obstacle avoidance one to safely achieve the task.

In order to select the appropriate controller, most of the methods relies on a fixed threshold (Cadenat et al., 2012). For example, an obstacle avoidance controller is triggered when the measured distance to an obstacle is smaller than a predefined value. Similarly, one uses a fixed distance to define a path around the obstacle which is safe for the robot. This approach has been efficiently used in the past to deal with static environment, but it does not seem to be fully appropriate to manage dynamic obstacles. For example, a moving object passing at the predefined distance of the robot will trigger the avoidance controller, although it might not represent any danger for the robot and the navigation could be achieved without avoiding it. Thus, in order to successfully perform a safe and efficient navigation, it is mandatory to consider more advanced methods to compute the parameters of the obstacle avoidance controller and to switch from one controller to the other. In this paper, it is proposed to extend our set of works presented in (Futterlieb et al., 2014; Durand-Petiteville et al., 2017; Leca et al., 2019) which are still restricted to static environments. The idea is to compute the residual value (Shi et al., 2005) between the predicted and measured distances and orientations to differentiate a static obstacle from a dynamic one. These values are then used to modify online the safety distance to the moving obstacle as well as the switching criteria. Thus, it becomes possible for the robot to efficiently avoid mobile obstacles.

The paper is organized as follows. Section II introduces the necessary preliminaries, namely the robot model, the spiral avoidance controller and the navigation controller. Section III and IV focus on the main contributions: the computation

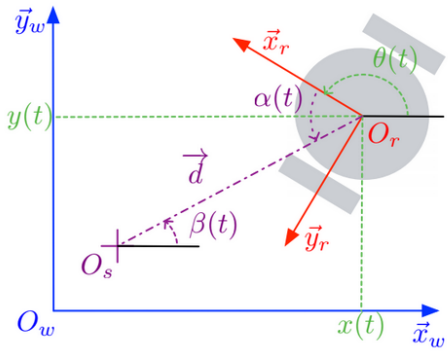

Figure 1: The robot model

of the adaptive distance and the strategy decision making. Section V presents the obtained simulation results. The paper ends with a conclusion and some prospects.

2 Preliminaries

2.1 The robot model

Here, we consider the differential robot presented in figure 1. It is equipped with a laser range finder and with classical localization sensors (IMU, odometry). Let define $F_w = (O_w, \overrightarrow{x}_w, \overrightarrow{y}_w, \overrightarrow{z}_w)$ and $F_r = (O_r, \overrightarrow{x}_r, \overrightarrow{y}_r, \overrightarrow{z}_r)$ the frames respectively attached to the world and the robot. The robot configuration is defined as $Q = [x(t)\ y(t)\ \theta(t)]^T$, where $x(t)$ and $y(t)$ are the coordinates of O_r in F_w, whereas $\theta(t)$ is the angle between $\overrightarrow{x}_w$ and $\overrightarrow{x}_r$ (see figure 1). The robot is controlled using the input vector $u(t) = [v(t)\ \omega(t)]^T$ where $v(t)$ is the linear velocity along $\overrightarrow{x}_r$, and $\omega(t)$ the angular velocity around $\overrightarrow{z}_r$. The other elements which appear in this figure (namely, $\vec{d}$, point O_s and angles α and β) will be described later.

2.2 The avoidance controller

A wide variety of techniques allow to perform obstacle avoidance in the literature. A very complete review can be found in (Hoy et al., 2015). The chosen approach is a reactive one, where no map is provided to the robot prior to the navigation. It is inspired by works where spirals are used to avoid obstacles (Mcfadyen et al., 2014; Futterlieb et al., 2014; Leca et al., 2019) or perform U-turn in agricultural fields (Durand-Petiteville et al., 2017). We briefly present the control law which will be used in this paper.

2.2.1 The spiral definition

Spirals have been studied in (Boyadzhiev, 1999). From this work, a spiral can be seen as the path described by a point O_p moving on a plane with respect to a fixed point O_s as shown in figure 2.

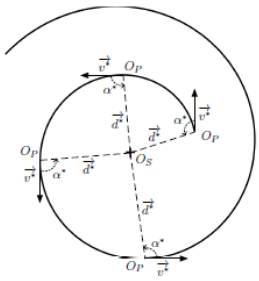

Figure 2: Model of a spiral

Two vectors can be introduced: $\vec{v^*}$ the velocity vector of the point O_p, and $\vec{d^*}$ the distance vector which connects the two points O_p and O_s. From this, $\alpha^*(t)$ can be defined by the angle between the two vectors $\vec{v^*}$ and $\vec{d^*}$. When $v^*(t)$ and $\alpha^*(t)$ are constant (that is, $v^*(t) = v^*$ and $\alpha^*(t) = \alpha^*$), point O_p follows a spiral whose center is defined by point O_s. The dynamics of d^* is defined by the following equation (Boyadzhiev, 1999):

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

This equation shows that the sense of motion and the spiral type (inward or outward) only depend on parameter α^*. Indeed, if $0 < \alpha^* < \pi$, O_p turns counter-clockwise with respect to O_s. If $0 < \alpha^* < -\pi$ it turns clockwise. The spiral is inward if $0 \leq \alpha^* < \pi/2$ or $0 \leq \alpha^* < -\pi/2$. It is outward if $\pi/2 < \alpha^* \leq \pi$ or $-\pi/2 < \alpha^* \leq -\pi$ Finally, if $\alpha^* = \pi/2$ or $\alpha^* = -\pi/2$, $d^*(t) = d^*(0)$. O_p then describes a circle of radius $d^*(0)$ around O_s.

The previous analysis shows the interest of using spirals to avoid obstacles. Indeed, let us consider again figure 1 and let us adapt the spiral to the robot model. To do so, we introduce $\overrightarrow{d} = \overrightarrow{O_s O_r}$ and $\alpha(t)$ the angle between $\overrightarrow{x}_r$ and $\overrightarrow{d}$. From the previous reasoning, it follows that, by making converging α and d towards α^* and d^*, it is possible to make the vehicle follow a dedicated spiral. α and d being given by laser data, this spiral is designed to guarantee obstacle avoidance. Now, it remains to design a control law allowing to make (α, d) converge towards (α^*, d^*). Such a controller has been proposed in (Durand-Petiteville et al., 2017).

2.2.2 The spiral controller design

As a first step, we express the dynamics of d and α (Durand-Petiteville et al., 2017):

$$\dot{d}(t) = -v(t) \cos(\alpha(t)) \tag{2}$$

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

A controller allowing to drive the robot towards the spiral is given by (Durand-Petiteville

et al., 2017):

$$\omega(t) = \lambda_s e_s(t) + \frac{v(t)}{d(t)} \sin(\alpha(t)) - \alpha_D * \dot{\epsilon}(t) \quad (4)$$

where the linear velocity v is fixed to a nonzero constant value, λ_s is a positive scalar vector. $e_s(t)$ is the error to be vanished. It is made of two separate terms respectively related to the error in α and d:

$$e_s(t) = \alpha(t) - \alpha^* - \alpha_D \epsilon(t) \quad (5)$$

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

$$\epsilon(t) = \text{sign}(d^*(t) - d(t)) \frac{min(||d^*(t) - d(t)||, n)}{n} \quad (6)$$

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 $d(t) = d^*(t)$. Additionnally, α_D is defined as:

$$\alpha_D = \begin{cases} \text{sign}(\alpha^*) * \pi - \alpha^* & \text{if } d^*(0) > d(0) \\ \alpha^* & \text{if } d^*(0) < d(0) \end{cases} \quad (7)$$

The control law (4) makes possible for the robot to converge towards any specific spiral (Durand-Petiteville et al., 2017). More details can be found in (Durand-Petiteville et al., 2017).

2.3 The navigation controller

This controller is intended to make the robot reach the given goal. Depending on the way it is defined, several solutions can be considered: visual servoing controller (Chaumette and Hutchinson, 2006), attractive potential field based controller (Ren et al., 2008), etc. Here, for the sake of simplicity, we have chosen to design a basic proportional controller to converge towards a goal O_g whose coordinates in the world frame are defined by $(x_G\ y_G)$ The idea is to correct the heading of the robot towards the goal, while maintaining a nonzero constant linear velocity v_0. The following control inputs allow to fulfill these objectives:

$$\begin{cases} v(t) &= v_0 \\ \omega(t) &= \arctan(\frac{y_G(t) - y(t)}{x_G(t) - x(t)}) - \theta(t) \end{cases} \quad (8)$$

Now that both controllers are available, it remains to define the adequate spiral parameters for a secure avoidance motion before presenting the navigation strategy.

3 Defining the spiral parameters

This section focuses on the spiral parameters allowing to guarantee a safe motion around the obstacle. They are three: the SCP (spiral center point), the angle α^*, and the distance d^*. The first two parameters have been defined following our previous works (Futterlieb et al., 2014; Leca et al., 2019). The SCP has been chosen as the closest point between the robot and the obstacle detected by the embedded lasers while angle α^* has been fixed to a constant value equal to $\pi/2$. The main difference arises with the definition of d^*. Indeed, it was kept constant in our above mentioned works, which was suitable while dealing with static obstacles only. To extend the method to dynamic scenes, it is necessary to consider a variable d^*. Our idea is to modify on-line its value when a mobile obstacle is encountered. More precisely, in this paper, it is proposed to dynamically compute the value of d^* thanks to the residual value between the predicted and measured distances between the robot and the SCP. From this residual, an adaptive threshold is computed (Shi et al., 2005), allowing to discriminate the static and dynamic obstacles and to adequately update d^* depending on the case. We first explain how to determine the residual before detailing the d^* value adaptation.

3.1 Computation of the residual

To compute the residual, the first step is to predict the distance between the robot and the obstacle at the current instant using previous laser informations. To state the problem, we consider two laser scans. The first one is acquired at the current time t_i and is denoted by ${}^{F_r(t_i)}P(t_i)$. It allows to determine the Spiral Center Point $O_c(t_i)$ (chosen as the closest detected point on the obstacle surface) and to extract the current measures of the distance $d(t_i)$ between O_r and $O_c(t_i)$ and the angle $\alpha(t_i)$. The second laser scan denoted by ${}^{F_r(t_p)}P(t_p)$ is obtained at a previous time $t_p < t_i$. Our goal is to use this latter laser scan to predict the value of the distance between the robot and the SCP at time t_i and compare it to its above measured value $d(t_i)$. To this aim, we first compute ${}^{F_r(t_i)}P(t_p)$ which denotes the set of points that would be perceived at time t_i if only the robot had been moving during the time interval $[t_p, t_i]$. To do so, the transformation matrix between two frames ${}^{F_r(t_i)}T_{F_r(t_p)}$ is expressed using local localization methods (i.e odometry, IMU)[1]. Thus, the predicted coordinates of the laser points from the previous acquisition in the current robot frame can be computed as follows:

$${}^{F_r(t_i)}P(t_p) = {}^{F_r(t_i)}T_{F_r(t_p)} \ {}^{F_r(t_p)}P(t_p) \quad (9)$$

From ${}^{F_r(t_i)}P(t_p)$, it is possible to compute the predicted closest point $O_c^{predicted}(t_i)$, and to deduce the predicted distance $d^{predicted}(t_i)$. From these

[1] As the time interval between t_p and t_i will not be chosen too large, the localisation drifting – intrinsic to local localisation methods – is considered to be negligible.

latter, we propose to define the residual $r(t_i)$ as follows:

$$r(t_i) = \frac{d^{predicted}(t_i) - d(t_i)}{t_p - t_i} \quad (10)$$

This residual value will help to decide whether the obstacle is static or not. Indeed, in the first case, $r(t_i)$ will be close to 0, whereas in the second case a greater value is expected. However, it might not be sufficient to obtain a robust discrimination because several phenomenons such as measurement noise, a too small motion of the mobile obstacle, modelling errors, etc. In this context, the definition of an adaptive threshold may be interesting to clearly classify dynamic and static objects.

3.2 Definition of the adaptive threshold and of the adaptive distance d^*

Adaptive threshold procedures summarized in (Emami-Naeini et al., 1988) or (Ding and Frank, 1991) are generally used to detect system faults and rely on a statistical study of the residual signal. See for example (Wang et al., 2003; Shi et al., 2005). In our case, the goal is to decide whether the encountered obstacle is static or dynamic, despite the above mentioned phenomenons. In (Shi et al., 2005), an adaptive threshold is computed from the mean and variance of the residual signal to detect faults in a nonlinear electro-hydraulic installation. Following a similar reasoning, we have computed the mean and variance of our own residue by considering its variation on q sample times:

$$\eta(t_i) = \frac{1}{q}\sum_{i=1}^{q} r(t_i), \quad (11)$$

$$\sigma^2(t_i) = \frac{1}{q-1}\sum_{i=1}^{q}(r(t_i) - \eta(t_i))^2. \quad (12)$$

From the above equations, the adaptive threshold δ at instant t_i is calculated as follows (Shi et al., 2005):

$$\delta(t_i) = \eta(t_i) \pm 2.17\sigma(t_i) \quad (13)$$

where the coefficient 2.17 corresponds to a 97% confidence level. Thus, if $r(t_i)$ belongs to the interval $[\delta^+(t_i), \delta^-(t_i)]$ where $\delta^+(t_i) = \eta(t_i) + 2.17\sigma(t_i)$ and $\delta^-(t_i) = \eta(t_i) - 2.17\sigma(t_i)$, then the obstacle is considered to be static. Otherwise, a dynamic obstacle is expected.

In addition, the upper bound of the adaptive threshold can be used to adjust the reference distance d^* as follows:

$$d^*(t_i) = d_{nominal} + \delta(t_i) \quad (14)$$

where $d_{nominal}$ is a suitable avoidance distance one would like to maintain in a static environment. This leads to a safer robot behavior. If a moving obstacle is going toward the robot, at each iteration, the real distance $d(t_i)$ is going to be smaller than the predicted one $d^{predicted}(t_i)$. Thus, the residual $r(t_i)$ will start increasing, and so will $\delta(t_i)$. Consequently, the desired distance $d^*(t_i)$ will increase, and the robot will avoid the obstacle at a greater distance than it would have done if the obstacle would have been static. Note that, in this particular use of the adaptive threshold, it may occur that $d^*(t_i)$ drops and reaches values which might be dangerous for the robot. To avoid such issues, a minimal bound corresponding to lowest acceptable distance has been added.

4 Navigation strategy

Now that the spiral parameters have been chosen, it remains to describe the navigation strategy. To define it, it is necessary to decide when switching between both controllers and which sense of motion is the most suitable. These two issues are addressed in the sequel.

4.1 Switching procedure

As presented before, two controllers may be applied to the robot: the navigation controller (or Go-To-Goal (GTG)) to reach the goal and the Spiral Avoidance controller (SA) when a collision risk occurs. To define the guarding conditions, the two following angles α_c and α_g are considered:
$\alpha = (\overrightarrow{x_r}, \overrightarrow{O_rO_c})$
$\alpha_g = (\overrightarrow{x_r}, \overrightarrow{O_rO_g}) = atan2(y_g - y, x_g - x) - \theta.$

They respectively provide the orientation of the robot towards the obstacle and towards the goal. The proposed guarding conditions are given as follows:

$$d_c < 2d^* \quad \text{and} \quad |\alpha_g - \alpha_c| < \pi/2 \quad (15)$$

Condition (15) is true if an obstacle lies within a $2d^*$ diameter half-circle heading toward the goal. In this way, the spiral avoidance controller is enabled only if an obstacle really lies on the robot trajectory towards the goal, leading to a true collision threat. Furthermore, because of the online modification of d^* when an dynamical obstacle is encountered, the robot is given some anticipation capacities. Indeed, if the obstacle is moving towards the robot, d^* will increase and condition (15) will be triggered earlier and so is the spiral avoidance controller.

Finally, note that the same conditions are used to switch from the navigation controller to the avoidance one and conversely. A smoothening procedure based on an averaging sliding window allows to avoid discontinuities in the control law applied to the robot at the switching time.

4.2 Sense-of-motion computation

When the obstacle avoidance controller is enabled, the robot has to choose its sense-of-motion (SOM) around the obstacle. In the case of static objects, the method consists in checking if it is on the right or on the left hand side of the robot. The following conditions are proposed: (Leca et al., 2019):

- If $\alpha \leq \alpha_g$, use a counter-clockwise SOM.
- Else if $\alpha > \alpha_g$, use a clockwise SOM.

These conditions appear to be limited in dynamic environments. Indeed, in such a scene, they could lead to a collision threat because they do not take into account the obstacles motion. To overcome this issue, we propose to use the residual on α, denoted by $\delta^{\alpha}(t)$. It is computed using the same method as the residual on the distance $\delta(t)$. It provides a basic knowledge of the direction of the obstacle motion with respect to the robot. In other words, it allows to determine whether the obstacle is coming from the left to the right relatively to the vehicle or the opposite. This information allows to compute a more suitable SOM:

- If $|\delta^{\alpha}(t)| \leq \delta^{\alpha}_{threshold}$, use the above mentioned conditions for static obstacles.
- Else if $\delta^{\alpha}(t) < -\delta^{\alpha}_{threshold}$, use a counter-clockwise SOM.
- Else if $\delta^{\alpha}(t) > \delta^{\alpha}_{threshold}$, use a clockwise SOM.

where $\delta^{\alpha}_{threshold}$ is a constant value chosen so that it can make the difference between static and dynamic obstacles. Additionally, the SOM is reset and reassessed if the robot switches back to the navigation controller, or if the distance between two consecutive SCP is greater than $2d^*$. This latter case occurs when the SCP has switched on another obstacle.

5 Simulation results

The method has been implemented using Matlab software. The considered environment is displayed in Figure 3. The robot starts from $O_r(0) = O_w$, and has to reach its goal $O_g = [33, 0]^T$. Its velocity has been set to $v^* = 1.5m.s^{-1}$. Between the robot starting position and the goal, lie three round obstacles. The two first ones denoted by (1) and (2) are dynamic, and respectively move at $1.5m.s^{-1}$ and $1.25m.s^{-1}$, while the last one (3) is static. The control gains and parameters are set as $\lambda_S = 1$, $n = 5$, and $d_{nominal} = 3m$. In order to simulate the noise coming from the laser range finder, an additive gaussian noise with $\sigma = 0.03$ has been added to the laser output. The main loop runs at a rate time $T_S = 0.02s$. To compute the residual, the interval between t_i and t_p has been chosen such as $t_i - t_p = 0.2s$, and the amount of samples used to compute the residuals in equation (12) is $q = 30$. To highlight the interest of

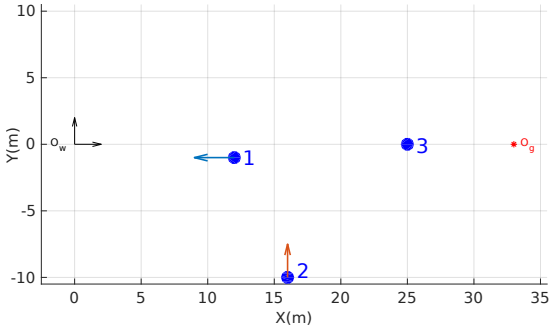

Figure 3: Simulated environment

our approach with respect to more classical ones, we have first implemented a navigation strategy based on a static d^* (no online adaptation of this distance). We present hereafter the corresponding results.

5.1 Results with a static d^*

In this case, we consider that $d^* = d_{nominal} = 3m$ all along the simulation. The threshold allowing to enable the avoidance is then set to $2d^* = 6m$. Furthermore, the SOM is chosen thanks to the method used in (Leca et al., 2019). Figures 4 and 5 respectively show the robot trajectory, its control inputs, the errors and the evolution of the distance. At $t = 2s$, obstacle (1) enters the $2d* = 6m$ safe zone around the robot, and the avoidance is triggered. Since at this time the obstacle is at the right of the robot, a clockwise SOM is selected. At $t \simeq 4.5s$, the obstacle is avoided and the GTG controller is applied. However, it can be seen that the robot has reached a distance close to 2 meters from the obstacle. At $t \simeq 6.5s$, the robot starts avoiding the second obstacle. At this time, this obstacle is at the right of the robot, but is moving from its right to the left. Hence, based on the static SOM condition, a clockwise SOM is chosen. The robot tries to move around the obstacle, but their velocities are close and the avoidance cannot be completed because the SOM cannot be reassessed. The robot "is then dragged" by the moving obstacle far from its goal, inducing a mission failure and highlighting the d^* constant approach limitations in a dynamic environment.

5.2 Results with the adaptive d^*

Here, we have simulated our approach based on the adaptive reference distance d^* combined with the new SOM choice condition. The minimal bound for d^* has been fixed to 2 m. Figures 6, 7 and 8 respectively present the robot trajectory, the control laws, the errors, the measured distance

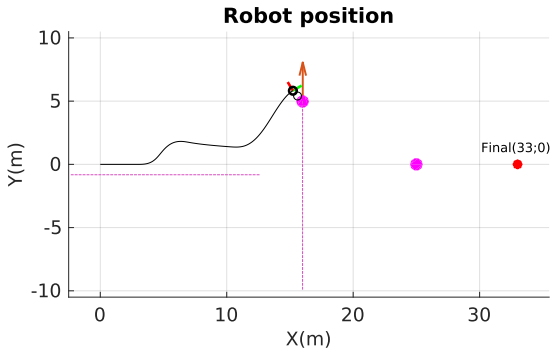

Figure 4: Robot (black line) and obstacles (pink dots) trajectories. [Static d^*].

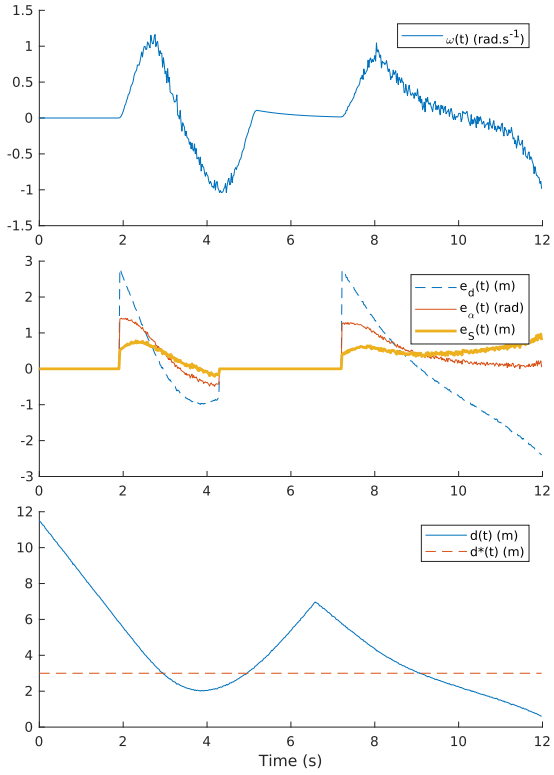

Figure 5: Angular control $\omega(t)$ – Errors $e_d(t)$, $e_\alpha(t)$, $e_S(t)$ – Distances $d(t)$ and $d^*(t)$ [Static d^*]

and its threshold, and the residual on angle α. They show that the task is successfully performed.

As the obstacle is moving toward the robot at a velocity of $1.5m.s^{-1}$, $d^*(t)$ quickly rises to around 4 meters. Thus, the trigger distance becomes close to $2d^*(t) \simeq 8m$, and is reached at $t \simeq 1s$. The avoidance controller is then enabled earlier than previously, which will insure a more efficient avoidance motion. Indeed, the distance between the obstacle and the robot does not drop below 3m instead of 2m in the previous case. As the residual δ^α belongs to the interval $[-\delta^\alpha_{threshold}, \delta^\alpha_{threshold}]$, the static algorithm is applied and a clockwise SOM is chosen. The first obstacle is successfully avoided. At $t = 7.5s$, the second obstacle is encountered and the avoidance is enabled anew. At this time, δ^α is negative and less than $-\delta^\alpha_{threshold}$. This means that the object is coming from right to left. Thanks to this information, the SOM is chosen as counter-clockwise, in order to move around it by the right hand side. At $t \simeq 12s$, obstacle (2) is successfully avoided and the navigation controller is applied to the robot until obstacle (3) is met. This latter being static, d^* remains constant and equal to $d_{nominal} \simeq 3m$. As $|\delta^\alpha| \leq \delta^\alpha_{threshold}$, a clockwise SOM is fixed. This obstacle (3) does not present any danger at $t \simeq 20s$, and the robot reaches its goal at $t \simeq 24s$. Finally, as it can be seen on figure 7, the spiral avoidance controller was able to bring $e_S(t)$ close to zero, while keeping the robot at a safe distance from the obstacle. The control law sent to the robot has remained in an acceptable range.

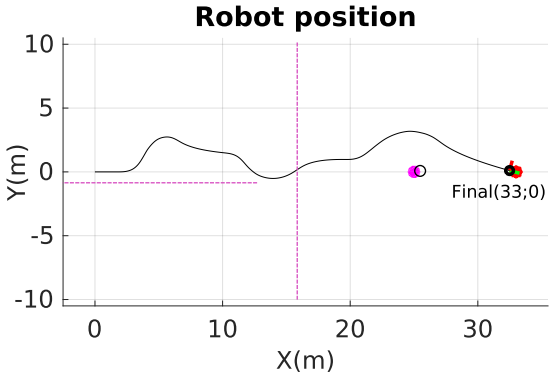

Figure 6: Robot (black line) and obstacles (pink dots) trajectories. [Adaptive distance]

6 Conclusion

This paper deals with the problem of navigating through a poorly known dynamic environment. The proposed strategy relies on two controllers allowing to reach the goal and to avoid both static and mobile obstacles. In order to cope with moving objects, the strategy has been enhanced with two features. The first one consists in adapting the value of the safe avoidance distance d^* during the mission whenever needed. To do so, an adaptive threshold has been computed thanks to the residual value of the difference between the predicted and current measured distances. The second contribution is related to the sense of motion which is now computed using a second dedicated adapted threshold allowing to take into account the obstacle motion. The proposed navigation strategy has been validated using Matlab. The simulation results show its relevance and efficiency to deal with both static and dynamic obstacles.

The proposed approach can be used with any kind of navigation controllers. It does not require any model of the environment and is particularly adapted to sensor-based navigation. For future works, we plan to experiment this solution on our

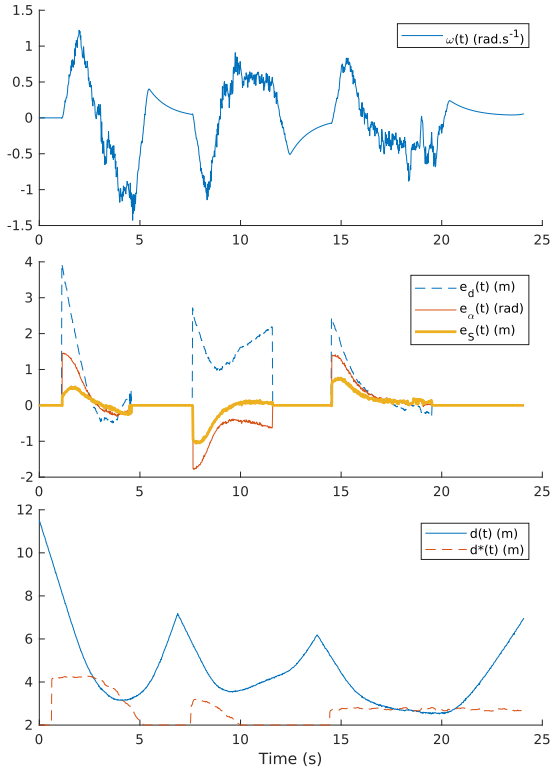

Figure 7: Angular control $\omega(t)$ – Errors $e_d(t)$, $e_\alpha(t)$, $e_S(t)$ – Distances $d(t)$ and $d^*(t)$ – Alpha residual $\delta^\alpha(t)$ [Adaptive distance]

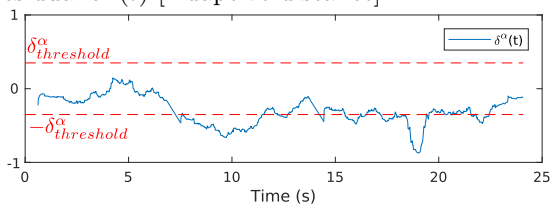

Figure 8: Alpha residual $\delta^\alpha(t)$

real robot Air-cobot which is intended to navigate through an airport to perform preflight inspection.

References

Boyadzhiev, K. N. (1999). Spirals and conchospirals in the flight of insects, *The college mathematics Journal* **30**(1): 23.

Cadenat, V., Folio, D. and Petiteville, A. D. (2012). Comparison of two sequencing techniques to perform a vision-based navigation task in a cluttered environment, *Advanced Robotics* **26**(5-6): 487–514.

Chaumette, F. and Hutchinson, S. (2006). Visual servo control, Part I: Basic approaches, *IEEE Robotics and Automation Magazine* **13**(4): 82–90.

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

Ding, X. and Frank, P. (1991). Frequency domain approach and threshold selector for robust model-based fault detection and isolation, *Proceedings of the IFAC symposium SAFEPROCESS*, Baden-Baden, Germany, pp. 307–312.

Durand-Petiteville, A., Cadenat, V. and Ouadah, N. (2015). A complete sensor-based system to navigate through a cluttered environment, *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Vol. 02, pp. 166–173.

Durand-Petiteville, A., Le Flecher, E., Cadenat, V., T., S. and S., V. (2017). Design of a sensor-based controller performing u-turn to navigate in orchards, *Informatics in Control, Automation and Robotics*.

Emami-Naeini, A., Akhter, M. M. and Rock, S. M. (1988). Effect of model uncertainty on failure detection: the threshold selector, *IEEE Transactions on Automatic Control* **32**(12): 1106–1115.

Futterlieb, M., Cadenat, V. and Sentenac, T. (2014). A navigational framework combining visual servoing and spiral obstacle avoidance techniques, *International Conference on Informatics in Control, Automation and Robotics*.

Hoy, M., Matseev, A. S. and Savkin, A. V. (2015). Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey, *Robotica* **33**: 463–497.

Leca, D., Cadenat, V., Sentenac, T., Durand-Petiteville, A., Gouaisbaut, F. and Flecher, E. L. (2019). Sensor-based obstacles avoidance using spiral controllers for an aircraft maintenance inspection robot, *European Control Conference (ECC)*.

Mcfadyen, A., A. and Mejias, L. (2014). Decision strategies for automated visual collision avoidance, *International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 715–725.

Ren, J., McIsaac, K. and Patel, R. (2008). Modified newton method applied to potential field navigation for nonholonomic robots in dynamic environments, *Robotica* **26**(1): 117 – 127.

Segvic, S., Remazeilles, A., Diosi, A. and Chaumette, F. (2007). A framework for scalable vision-only navigation, *Advanced Concepts for Intelligent Vision Systems, ACIVS'07*, Delft, The Netherlands, pp. 112–124.

Shi, Z., Gu, F., Lennox, B. and A.D., B. (2005). The development of an adaptive threshold for model-based fault detection of a nonlinear electro-hydraulic system, *Control Engineering Practice* **13**: 1357–1367.

Wang, X., Kruger, U. and Lennox, B. (2003). Recursive partial least square algorithms for monitoring complex industrial processes, *Control Engineering Practice* pp. 613–632.