Sensor-based algorithm for collision-free avoidance of mobile robots in complex dynamic environments

Dimitri Leca, Viviane Cadenat, Thierry Sentenac

To cite this version:
Dimitri Leca, Viviane Cadenat, Thierry Sentenac. Sensor-based algorithm for collision-free avoidance of mobile robots in complex dynamic environments. 2019 European Conference on Mobile Robots (ECMR), Sep 2019, Prague, Czech Republic. p.1-6, 10.1109/ECMR.2019.8870344 . hal-02334205

HAL Id: hal-02334205
https://hal-mines-albi.archives-ouvertes.fr/hal-02334205

Submitted on 2 Mar 2020

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.
Sensor-based algorithm for collision-free avoidance of mobile robots in complex dynamic environments

D. Leca\textsuperscript{1,2}, V. Cadenat\textsuperscript{1,2}, T. Sentenac\textsuperscript{1,3}

Abstract—This paper deals with the problem of navigation of unmanned vehicles through poorly known environments cluttered with static and dynamic obstacles. The robot is equipped with a LiDAR able to provide a scan of the surroundings and with classical dedicated localization sensors (odometry, IMU). The proposed navigation strategy relies on: (i) a new concept called Enhanced Laser Scan (ELS), which is built from the current laser scan by adding virtual points along the predicted trajectory of the obstacles; (ii) two sensor-based controllers allowing respectively to reach the goal and to avoid obstacles. These controllers relying on the richer information provided by the ELS, they will be able to anticipate and safely avoid both static and moving obstacles; (iii) a high-level decision process allowing a better choice of the sense-of-motion (SOM) around the obstacle and its reassessment if needed.

I. INTRODUCTION

In mobile robotics, one of the key feature is the ability for the robot to safely and smoothly navigate through an unknown environment. In this environment, the robot will encounter static as well as dynamic obstacles. If an extensive amount of papers has been published about how to deal with static obstacles \cite{1}, the case of dynamic obstacles is still a discussed issue \cite{2}. Indeed, in an unknown environment, few assumptions can be made about the obstacles lying in the robot vicinity. These obstacles can be rigid or with changing shapes. Their trajectory and velocity can be subject to unpredictable changes with time. In these cases, global planners are limited \cite{1}, since they require a previous knowledge of the environment. Because of this minimal representation, local approaches appear to be more suitable, since they will allow the robot to reactively adapt to any change in its environment. They can be split in two categories: local planners, that use the local information provided by the sensors to compute a short-horizon trajectory, and reactive controllers, that directly generate the current control using the current provided information.

Among local planners that take into account dynamic obstacles, the well-known Dynamic Window Approach (DWA) has been generalized in \cite{3} and \cite{4} to handle moving obstacles. The concept of Velocity Obstacles (VO) chooses among all possible velocities the one that guarantees non-collision. This method has been extended to handle dynamic obstacles, with the Acceleration-Velocity Obstacles \cite{5} or the Non-Linear Velocity Obstacles \cite{6}, \cite{7}. However, all of these methods assume that the encountered obstacles are small and cylindrical to take into account their velocities. The methods based on Artificial Potential Fields have also been widely investigated to deal with mobile obstacles. They suffer from well-known drawbacks, such as local minima \cite{8}. As local planners, their adaptation to dynamic environments are often performed under strong assumptions, such as: the precise knowledge of the velocity and the acceleration of each obstacle, their size and shape, or the number to be avoided simultaneously \cite{9}, \cite{10}, \cite{11}, \cite{12}. Another class of methods, called bug methods, consists in the following reasoning: moving toward the goal following a straight line, switching to an avoidance method if a collision threat occurs, and resuming to a straight line toward the goal once the obstacle is avoided. These methods require few assumptions and knowledge about the environment. They have been successfully applied to different kinds of robots in \cite{13}, \cite{14}, \cite{15}, and have dealt with complex scenes involving numerous obstacles whose shapes and movements have evolved during the mission.

In this work, we assume that the environment is poorly known: no map is provided prior to the navigation and no information about the position, shape or motions of the possibly encountered obstacles is available. The robot is equipped with a LiDAR able to provide a scan of the surroundings. In such a case, following our previous study \cite{16}, local reactive methods have been shown to be more adapted. It is proposed such a navigation strategy based on two sensor-based controllers allowing respectively to reach the goal and to avoid static obstacles. Here, the presented strategy extends our previous works on that topic \cite{17}, \cite{18}, \cite{16}. In this set of works, the avoidance technique relies on the definition of a spiral \cite{19} as first proposed in \cite{20} and \cite{21} for UAV (Unmanned Aerial Vehicles). If our most recent strategy developed in \cite{16} allows to safely and efficiently navigate through a static environment cluttered with various shaped obstacles, it is still impossible to deal with dynamic ones. This paper allows to overcome this limitation by proposing an enhanced navigation strategy able to avoid both static and dynamic objects whose both motion and shape are not a priori known. Three main improvements have been made with respect to \cite{16}. At low level, the points provided by the LiDAR have been treated so that: (i) static and dynamic objects can be separated; (ii) a prediction of the moving objects velocity is performed; (iii) this prediction is used to add to the current LiDAR scan a set of virtual points representing each obstacle predicted motion. This allows to produce an ‘enhanced laser scan’ (ELS) which will be

\textsuperscript{1} CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
\textsuperscript{2} Univ de Toulouse, UPS, LAAS, F-31400, Toulouse, France
\textsuperscript{3} Institut Clément Ader (ICA), Université de Toulouse, CNRS, Mines Albi, UPS, INSA, ISAE-SUPAERO, Campus Jarlard, F-81013 Albi CT Cedex 09, France
used to feed the avoidance controller designed in [16]. The
same control law is then applied to the robot but, thanks
to the richer information contained in the ELS, it becomes
possible to anticipate the motion of mobile obstacles. The
second improvement with respect to [16] is a high level
one. It concerns the choice of the sense of motion (SOM)
around the obstacle. The SOM decision takes into account
the motion of the obstacle and can be reassessed if this
latter happens to suddenly modify its trajectory. This allows
to avoid undesirable behaviors where the robot could be
dragged away by an object moving at the same speed.
Finally, a linear velocity profile has been designed to improve
the robot behavior in the obstacle vicinity.

The paper is organized as follows. Section II introduces
the spiral model, its parameters, and the control laws designed
to follow the spirals. Section III introduces the algorithms
designed to take into account dynamic parts of the envi-
ronment. Section IV deals with the implemented decision
process while section V presents simulation results showing
the interest of the approach.

II. PRELIMINARIES

This section deals with the preliminaries of the spiral
avoidance technique [16].

A. Spiral modelling

The section focuses on parameters used to define the
equational 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. This point is considered as the center
of the spiral. \(\vec{v}^*\) is the velocity vector applied to \(O_p\) and its
 norm is denoted \(v^*\). Moreover \(\vec{d}^2\) is the vector connecting
\(O_s\) to \(O_p\) whose norm is \(d^*\). Finally \(\alpha^* (t)\) is defined as
the oriented angle between \(\vec{v}^*\) and \(\vec{d}^2\). In [19] 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 \(\alpha^*\) in the sequel. Moreover [19] states
that \(d^*(t) = -v^* \cos(\alpha^*)\). Thus the executed spiral only
depends on the value of \(\alpha^*\). Its sign allows to define the SOM
(clockwise if negative, anticlockwise if positive), while its
value fixes the type of spiral: inward if \(|\alpha^*| < \pi/2\), outward
if \(\pi/2 < |\alpha^*| \leq \pi\), 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 \(\alpha^*\) and \(d^*\) allows us first to choose the SOM 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.

B. Robot modelling

The robot has four-wheel skid steering drive, which allows
the robot to turn on itself. Its model is presented in figure 1b.
\(F_w = (O_w, \vec{x}_w, \vec{y}_w, \vec{z}_w)\) is the frame linked to the world,
while \(F_r = (O_r, \vec{x}_r, \vec{y}_r, \vec{z}_r)\) 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 \(\vec{x}_w\)
and \(\vec{x}_r\). \(O_s\) is the center of the spiral to be followed, while
\(\vec{d}\) represents the vector connecting \(O_s\) to \(O_r\) and \(\alpha(t)\) is
the angle between \(\vec{x}_r\) and \(\vec{d}\). \(\beta(t)\) is the angle between \(\vec{x}_w\)
and \(\vec{d}\). Knowing that \(\alpha(t) = \pi - \theta(t) + \beta(t)\), it should be
noticed that:

\[
\dot{\alpha}(t) = -\dot{\theta}(t) + \dot{\beta}(t) = -\omega(\theta(t) + v(t) \sin(\alpha(t)) \quad (1)
\]

Consequently, \(d(t)\) represents the distance between \(O_s\) and
\(O_r\). Following [19]:

\[
\dot{d}(t) = -v(t) \cos(\alpha(t)).
\]

C. Obstacle avoidance controllers design

To perform the avoidance, the robot must first reach the
desired spiral, and then follow it around the obstacle. Two
two controllers have been proposed in [16].

1) Definition of the errors: 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 \(\alpha^*\), it is proposed to impose \(v(t) = v^* \neq 0\).

2) First controller design: The following control law \(\omega_A\)
allows to make the errors \(e_\alpha\) and \(e_d\) converge toward zero
asymptotically thanks to an exact input to state linearization
method [22]:

\[
\omega_A(t) = \lambda_1 e_d(t) + \lambda_2 e_\alpha(t) + v^*(\sin(e_\alpha(t) + \alpha^*)) + e_d(t) + d^*(t)\quad (2)
\]

This controller maintains the robot along the desired spiral,
with \(\lambda_1, \lambda_2 > 0\), but suffers from singularities when \(e_\alpha(t) + \alpha^* = \alpha(t) = k\pi, k \in \mathbb{Z}\).

3) Second controller design: This controller is intended to
avoid this latter problem. Following [18], an hybrid error
is introduced: \(e_S(t) = e_\alpha(t) - \epsilon(t)\alpha_D\), where \(\epsilon(t)\) is the
normalized error between \(d^*(t)\) and \(d(t)\), saturated to \(\pm 1:\)

\[
\epsilon(t) = \frac{\min(||d^*(t) - d(t)||, n)}{n}\quad (3)
\]

where \(n \in \mathbb{N^*}\). Additionally, \(\alpha_D\) is defined as \(\text{sign}(\alpha^*) \pi - \alpha^*\)
if \(d^*(0) > d(0)\), or \(\alpha^*\) if \(d^*(0) < d(0)\). To make \(e_S(t)\) vanish,
the following controller \(\omega_B\) is proposed in [18]:

\[
\omega_B(t) = \lambda_S e_S(t) + \frac{v(t)}{d(t)} \sin(\alpha(t)) - \dot{\alpha}(t, d)\quad (4)
\]

where \(\lambda_S > 0\). As shown in [18], this controller is locally
asymptotically stable, once \(\alpha(t)\) overpasses \(\alpha^*(t)\).
4) **Switching strategy:** To overcome the singularity problem which occurs when \( \alpha(t) \) vanishes, it is proposed to switch between \( \omega_A \) and \( \omega_B \). When the error \( e_{\alpha}(t) \) drops below a threshold \( e_{\alpha}^{\text{switch}} \), the \( \omega_A(t) \) controller is sent to the robot. Else, \( \omega_B(t) \) is sent. To obtain the final control law \( \omega(t) \), a sliding-window smoothing is applied to avoid discontinuities in the control inputs when switching between \( \omega_A \) and \( \omega_B \).

D. **Spiral-based navigation strategy for static environments**

1) **Choice of \( \alpha^* \) and \( d^* \):** In [16], an angle of \( \alpha^* = \pm \pi/2 \) is imposed. Its sign depending on the SOM around the obstacle. Consequently, \( d^* \) is constant.

2) **Choice of the SCP:** After a LiDAR acquisition around the robot, the closest point \( O_e \) from the robot is computed. Then, all the LiDAR points within a \( 2d^* \) meters radius circle centered on \( O_e \) are used to compute their barycenter \( O_h \). Finally, \( O_e \) and \( O_h \) are compared, and the closest one to the robot is chosen as the spiral center point \( O_s \).

3) **Go-to-Goal controller and switching conditions:** The Go-To-Goal (GTG) controller can be defined by any controller allowing to reach the goal. Here, we have chosen a basic proportional controller correcting the heading of the robot to reach \( O_g \). As explained in [16], the switch between the avoidance controller and the GTG controller occurs if an obstacle lies within a half-circle heading toward the goal, whose radius is between \( d^* \) and \( 2d^* \) depending on the relative orientation between the robot and the obstacle. Thus, if an obstacle lays in front of the robot, the avoidance will be triggered earlier.

This section has been devoted to a brief recalling of the navigation strategy described in [16]. This method has been proven to be efficient for static environments but cannot deal with mobile obstacles. This paper extends these works by proposing three improvements: (i) the design of an enhanced laser scan (ELS) containing the current laser points and the virtual points modelling the mobile obstacle motion; (ii) the possible reassessment of the SOM to react adequately with respect to obstacle motion; and (iii) the design of a safer linear velocity profile. Thanks to these improvements, the same control law can be used for both static and dynamic environments, generalizing our previous works.

III. **ENHANCED LASER SCAN**

This section focuses on our first contribution, the computation of the dynamics of each point detected by the LiDARS, and the generation of the artificial laser points. The aim of these virtual points is to model the expected evolution of the dynamic parts of the environment. These virtual points will be merged with the initial laser scan to provide an enhanced laser scan (ELS) that will feed the algorithm presented in part II. Using the ELS, the robot will anticipate the evolution of each moving obstacle, taking safer decisions such as an earlier avoidance trigger or a more suitable SOM.

A. **Detection of the moving points**

We denote by \( R_p \) and \( R_e \) the frames attached to the robot at time \( t_p \) and \( t_e \), with \( t_p < t_e \). We also introduce \( P_{R_e}(t) \) and \( P_{R_p}(t) \) the arrays of laser point coordinates taken at time \( t \), given in the robot frames \( R_p \) and \( R_e \). The algorithm works by comparing the points from the current laser scan \( P_{R_e}(t_e) \) taken at time \( t_e \) with the points from a previous laser scan \( P_{R_p}(t_p) \). The transformation matrix \( R_e^T R_p \) between \( R_p \) and \( R_e \) can be obtained using local localization methods. Thus, the coordinates of the points from the previous acquisition in the current robot frame \( R_e \) can be computed such as:

\[
P_{R_e}(t_p) = R_e^T R_p \cdot P_{R_p}(t_p)
\]

Algorithm 1 finds for each point in \( P_{R_e}(t_e) \) its closest point in \( P_{R_p}(t_p) \). If this distance is over a threshold \( D \), the point is considered as moving. Else, it is not. The algorithm produces two output matrices, \( P_{R_e}(t_e) \) and \( P_{R_p}(t_p) \). These matrices respectively contain the subset of points from \( P_{R_e}(t_e) \) and \( P_{R_p}(t_p) \) that are seen to be moving, and will be used to characterize the robot environment.

**Algorithm 1 Detection of the moving points**

\[
P_{R_e}(t_e), \ P_{R_p}(t_p) \leftarrow []
\]

for all Point P1 in \( P_{R_e}(t_e) \) do

\[ D_{\text{min}} \leftarrow +\infty \]

for all Point P2 in \( P_{R_p}(t_p) \) do

\[ D_{\text{min}} \leftarrow \min(D_{\text{min}}, \text{distance}(P1,P2)) \]

end for

if \( D_{\text{min}} > D \) then

\[ P_{R_e}(t_e) \leftarrow [P_{R_e}(t_e); P1] \]

\[ P_{R_p}(t_p) \leftarrow [P_{R_p}(t_p); P2] \]

end if

end for

B. **Characterization of the obstacles**

1) **Obstacle clustering:** A spatial clustering is operated on the points contained in \( P_{R_e}(t_e) \). Since the obstacles are well-separated, any well-known clustering technique, such as hierarchical clustering [23] can be used. This allows to be able to handle any number of obstacles.

2) **Obstacle velocity vector computation:** After the clustering, the velocity vector of each obstacle is estimated. For each cluster, the barycenters \( B_{R_e}(t_e) \) and \( B_{R_p}(t_p) \) of their corresponding points in \( P_{R_e}(t_e) \) and \( P_{R_p}(t_p) \) are computed. The velocity vector for this cluster is obtained as follows:

\[
\bar{V}_{R_e}(t_e) = \begin{pmatrix} \bar{v}_x(t_e) \\ \bar{v}_y(t_e) \end{pmatrix} = \frac{B_{R_e}(t_e) - B_{R_e}(t_p)}{t_e - t_p}
\]

C. **Projection of the virtual points**

Using the velocity vector, the points that will be added to the ELS are computed.
1) Horizon: It has been seen that the avoidance can be triggered at a maximal distance of $2d^*$. Depending on its velocity $v(t)$, the robot could reach this distance in $2d^*/v(t)$ seconds. We then define $t_h = 2d^*/v_{\text{max}}$. At any time when the avoidance could be triggered, it is necessary to take into account where any moving obstacles might be in $t_h$ seconds. Consequently, depending on its cluster, each moving point will be projected along the vector $\vec{d}_n = \vec{V}_{R_n}(t_c) \cdot t_h$.

2) Projection: For each moving obstacle, virtual points are added to the initial laser scan to model the future trajectory of this obstacle. The method consists in translating, for each cluster, the points from a given distance along the velocity vector, and to add a hull linking the initial points and the translated points. Furthermore, in order to prevent the angular control law to unexpectedly reach too high values, no virtual point should be added within a $d^*$ perimeter around the robot. Figure 2 shows how the virtual points are computed. The environment is cluttered with two obstacles. One rectangular static obstacle (A), and one circular dynamic obstacle (B). The solid lines represent both obstacles at $t = t_c$. The solid blue points are the points $P_{R_c}(t_c)$ currently detected by the LiDAR. The dotted circle represents the (B) obstacle as it was at time $t = t_p$, while the grey star points are the points $P_{R_c}(t_p)$, that were detected at the previous time $t_p$, projected in the current robot frame $R_c$. At the beginning of the algorithm, each point from $P_{R_c}(t_p)$ is matched with its closest point in $P_{R_c}(t_c)$, represented by the green arrows. If the distance between these matched points is bigger than a threshold, these points are considered to be moving. This is the case of the points lying in the circle, but not the case of those belonging to the static obstacle (A). At the end of this process, $P_{R_c}(t_p)$ contains the five grey star points, $P_{R_c}(t_c)$ the five blue points lying in the circle. Then, the clustering is performed, and the barycenters $B_{R_c}(t_c)$ and $B_{R_c}(t_p)$ are computed, represented respectively by the red circle point and the red star point. $\vec{V}_{R_c}(t_c)$ is then calculated, represented by the black arrow between the red points. This process being done, the virtual points can be computed and added to the final scan. The five blue round points are translated along their respective vector $\vec{d}_n$, generating the five squared blue points. The hull is then created, producing the blue star points.

IV. Navigation strategy

A. Sense of motion computation

For static obstacles, the condition is based on the sign of $\alpha_c$: if $\alpha_c \leq 0$, select a clockwise SOM. Else, select a counter-clockwise SOM. For a dynamic environment, the future position of the obstacle must be taken into account at the decision instant. Typically, if the obstacle is at the right of the robot, and moving from the right to the left, the previous computation would produce a clockwise SOM. Such a SOM would make the robot trajectory to cross the obstacle expected path, leading to a collision threat. The proposed algorithm anticipates the motion of dynamic obstacle. When an obstacle is detected and the switch between the GTG controller and the SA controller is triggered, the SOM is determined using $O_c$ and its associated velocity vector $\vec{V}_{R_c}(t) = \begin{pmatrix} v_x(t) \\ v_y(t) \end{pmatrix}$.

- If $v_y = 0$, apply the conditions for static obstacles.
- Else if $v_y > 0$, select a counter-clockwise SOM. It means that the obstacle is going from right to left.
- Else if $v_y < 0$, select a clockwise SOM. It means that the obstacle is going from left to right.

B. SOM reassessment

In complex cases (multiple obstacles, unpredictable obstacle dynamics), it might happen that the SOM computed when the avoidance process was first triggered makes the robot be dragged away by an obstacle moving at the same speed than the robot. Consequently, it is necessary to detect these situations and reassess the SOM whenever needed. These cases can be detected by checking if $v_x > 0$ and $|v_y| < v_{\text{min}}$. If these conditions are fulfilled, the robot is going in the same direction than the obstacle. If this case occurs, the algorithm reassesses the SOM. Furthermore, at each iteration, the distance between two successive SCP is computed. If this distance is greater than a given threshold, the new SCP belongs to a new obstacle. In this case, the SOM is also reassessed.

C. Linear velocity

In [16], the robot was operating at the same constant linear velocity $v$. For a safer behavior, the robot linear velocity $v$ is mapped with $e_{\alpha}(t)$, to evolve between a maximum velocity $v_{\text{max}}$ and a minimum velocity $v_{\text{min}}$:

$$v(t) = v_{\text{max}} - (v_{\text{max}} - v_{\text{min}}) \frac{|e_{\alpha}(t)|}{\pi/2}$$

(7)

This formula leads to a safer behavior. When an obstacle lies in front of the robot ($|e_{\alpha}(t)| \approx \pi/2$), the velocity will be minimal. If it is on the side of the robot ($|e_{\alpha}(t)| \simeq 0$), a maximal linear velocity will be produced.

V. Simulation and results

A. Simulation description

We have evaluated our method using Matlab software. Starting from its initial pose $O_r(0) = O_w$, it has to reach a goal $O_g$ whose coordinates in the initial robot frame are
The obstacles at obstacles. Figure 3 shows the position and the velocity of \([x_g, y_g] = [30, 0]\). Between the robot and the goal lie several obstacles. Figure 3 shows the position and the velocity of the obstacles at \(t_1 = 0s, t_2 = 7.5s\) and \(t_3 = 12s\). \(O_1\), \(O_2\) and \(O_3\) are three small convex obstacles, which could represent humans walking. \(O_2\) changes its trajectory at \(t_2\). \(O_4\) is a car-sized obstacle, and starts moving at \(t_3\). \(O_5\) is a static obstacle. This environment aims at simulating a typical human environment. The robot linear velocity \(v(t)\) evolves between \(v_{min} = 0.5ms^{-1}\) and \(v_{max} = 1.5ms^{-1}\). To ensure a safe avoidance motion, \(d^* = 3m\). For the controllers, the gains are set as: \(\lambda_1 = 0.2\), \(\lambda_2 = 0.2\) and \(\lambda_S = 0.5ms^{-1}\). \(e_{alpha^{switch}} = \pi/12\), \(n = 5\), and \(v_0^y = 0.5ms^{-1}\). An additive white gaussian noise with \(\sigma = 0.03\) is applied to the LiDAR distance output. The algorithm runs with a rate time \(T_s = 0.02s\). Furthermore, the interval between \(t_c\) and \(t_p\) is chosen at 10 iterations, hence \(t_c - t_p = 0.2s\). To smoothen the control, an averaging sliding window of 5 iterations is applied.

B. Results

The robot succeeds in reaching its goal in 28.2 seconds, safely avoiding all the obstacles. Figure 4 displays the pose of the robot and its trajectory at four key moments. Red points represent virtual laser points added to the scan. They can be compared with the real obstacle velocities shown on figure 3 and represented by arrows. It can be seen that at each iteration, the virtual laser points are added along the expected future path of each obstacle. The black dots represent the current SCP. Figure 5 displays the linear velocity \(v(t)\) and the angular velocity \(\omega(t)\) as well as the type of used controllers. Figure 6 presents the evolution of \(e_d(t)\) and \(e_{alpha}(t)\), as well as \(d(t)\) and \(d_{alpha}(t)\), that represents the closest physical point to the robot.

1) Avoidance of \(O_1\): From \(t = 0s\) to \(t = 1s\), the robot uses the GtG controller and goes toward the goal. \(O_1\) and \(O_2\) are within LiDAR detection range. The algorithm detects that these obstacles are moving, and adds virtual points along their future predicted trajectories. The closest point \(O_2\) is computed and belongs to the virtual points added from \(O_1\). This point being within the avoidance trigger range, the robot switches from the GtG controller to the SA controller. At this point, thanks to the obstacle velocity vector, the algorithm detects that the obstacle is going from right to left, and decides a c.c.w. SOM, as it can be seen on figure 4a.

2) Avoidance of \(O_2\): At \(t = 6s\), the SCP becomes the closest point \(O_2\) which is a virtual point derivated from \(O_2\). Such a jump in the SCP position triggers the reassessment of the SOM. Because \(O_2\) goes from left to right, a c.w. sense of motion is chosen (Figure 4b). As shown in figure 3, \(O_2\) suddenly changes its trajectory at \(t = 7.5s\). Because of its initial choice of SOM, the robot was trying to avoid it in the same direction it is now going. The condition presented in IV-B is met, and the SOM is reassessed, leading to a new c.w. sense of motion, as shown on Figures 4c. and 4d. A pike in the angular control output \(\omega(t)\) can be observed on Figure 5, corresponding to the SOM changing.

3) Avoidance of \(O_3\) and \(O_4\): At \(t = 16s\), the obstacle \(O_3\), which is moving from the right to the left with respect to the robot. A c.c.w. SOM is fixed. Soon after, \(O_4\) becomes the closest detected obstacle. This jump in the SCP leads to a reassessment of the SOM, chosen as c.w., since the obstacle is moving from left to right.

4) Avoidance of \(O_5\): At \(t = 20s\), the mobile obstacle \(O_4\) has been avoided, and the robot arrives in front of the static objet \(O_5\). A c.c.w. SOM is chosen. At \(t = 28.2s\), the simulation ends with the robot reaching its goal.

This simulation highlights the advantages of our method. Using the information about environment provided by the enhanced laser scan, the robot is able to take better decisions about the SOM, leading to a quicker and safer avoidance. Moreover, it can also handle a sudden modification of the obstacles direction by switching its SOM. It can be seen on figure 6 that in spite of the numerous dynamic obstacles, no collision occurs. The controls \(v(t)\) and \(\omega(t)\) also remain continuous and within acceptable ranges.

VI. CONCLUSION

This article has presented a novel strategy allowing a mobile robot to navigate through a dynamic environment. It extends previous works based on the spiral avoidance method. It combines the control laws proposed in [16] with a novel method called enhanced laser scan, where virtual points are added to the laser scan. A new decision layer has also been added to improve the choice of the sense-of-motion. This complete navigation strategy has been tested on Matlab software, and has been proven to be efficient. Therefore, our next step is to implement it on the Air-Cobot plateform in LAAS.

REFERENCES

**Fig. 4**: Obstacle position and robot position and trajectory at different times

**Fig. 5**: Linear velocity (in $\text{ms}^{-1}$), angular velocity (in $\text{rad.s}^{-1}$) and controller used (GtG, SA1, or SA2)

**Fig. 6**: Distance error $e_d(t)$, angular error $e_o(t)$, distance to the SCP $d(t)$ and distance to the closest physical point $d_o(t)$.


