Robotics and Autonomous Systems 88 (2017) 51–70

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Reactive versus cognitive vehicle navigation based on optimal local and global PELC*

Lounis Adouane

Clermont Auvergne University, Institut Pascal, UCA/SIGMA - UMR CNRS 6602, France

ARTICLE INFO

Article history:
Received 19 October 2015
Accepted 4 November 2016
Available online 16 November 2016

Keywords:
Autonomous vehicle navigation
Hybrid (reactive/cognitive) control architecture
Obstacle avoidance
Limit-cycle approach
Local and global path planning
Multi-criteria optimization
Optimal planning

ABSTRACT

This paper addresses the challenging issue of determining the most suitable control strategy (planning–decision–action and their interactions), for autonomous navigation of vehicles which must deal with different environments contexts (e.g., cluttered or not, dynamic or not, etc.). The paper's main proposals are decomposed into two main parts: Firstly, the proposition of reliable and flexible components to perform short and long-term planning: at beginning, a generic and safe path planning-based on Parallel Elliptic Limit-Cycle (PELC) and its multi-criteria optimization (PELC*) have been proposed to perform either reactive or cognitive navigation. Afterward, it is proposed to suitably sequence several PELC/PELC* in order to obtain an optimal global path-based on PELC (gPELC*). Secondly, this paper proposes an overall Hybrid (reactive/cognitive) multi-controller architecture for autonomous navigation using PELC* and gPELC*. This architecture has been designed in order to use a uniform set-points convention and a common control law to perform several sub-tasks (e.g., obstacle avoidance, target reaching/tracking, path following, etc.). A multitude of simulations and a real experiment have been performed in order to confirm the potentialities of the overall proposed methodology.

1. Introduction

To perform fully autonomous robot navigation, while having accurate perception and localization capacities [1–3], the robot must also have the ability to be controlled online in different kinds of environments (e.g., cluttered or not, dynamic or not, uncertain or not, etc.) and to react safely to unpredictable events. Thus, the used control architecture must permit us to answer this important question "*How do we reach safely and efficiently a predetermined location in an environment while taking into account available environmental knowledge (the road limits for instance) and reacting online to unpredictable events (e.g., other robots, obstacles, etc.)?*".

Furthermore, it is not sufficient to guarantee only the reliability and the safety of the navigation; the robot must also ensure, in transportation applications for instance [4,5], smooth navigation for the comfort of the passengers. In [6], the author characterizes this smooth navigation while using a cost function which reflects the trade-off between the travel time and the integral of acceleration (which characterizes the jerking amount of angular and linear robot velocities). Fully autonomous navigation needs therefore to satisfy simultaneously a multitude of criterion. For this aim it is important to have a reliable, safe and flexible control architecture [7]. Several navigation strategies (using dedicated control architectures) have been proposed in the literature. They permit autonomous navigation even in dynamic and cluttered environments. This means that "obstacle avoidance" function is always an important primitive and is tightly inherent to the performed autonomous navigation strategy. Thus, special attention should be taken for its development [7]. The generic proposed obstacle avoidance primitive will be detailed in Section 3.1.1.

1.1. Reactive versus cognitive control architecture

Control architectures can be split into two categories: Cognitive and Reactive. The cognitive (or deliberative) architectures make their main focus on the path/trajectory[1] planning and re-planning [8], while generally taking into account the overall environment knowledge. The obtained trajectory takes into account all obstacle configurations (and maybe their dynamic) in the planning step. In fully cognitive navigation, once a trajectory is obtained, the robot follows it as accurately as possible using the dedicated or generic control laws, for instance using the well-known laws proposed in [9] or [10]. A multitude of methods exist in the literature to deal with path/trajectory planning, among them: Artificial Potential Field (APF) [11]; Voronoï diagrams [12]; visibility graphs [13]; navigation functions [14] or planning based on grid map [2]; Rapidly-exploring Random Tree (RRT) [15], Sparse A* Search (SAS) [16]. It

[1] It is to be noted that the term trajectory or path are used according, respectively, if the time is taken or not into account during the planning phase.

E-mail address: Lounis.Adouane@univ-bpclermont.fr.

http://dx.doi.org/10.1016/j.robot.2016.11.006
0921-8890/

is commonly used in cognitive control architectures a pre-planned reference trajectories, which means that they are properly selected before robot movement [17]. The majority of the cited techniques could be used even for short or long-term path/trajectory planning. In the first case, these techniques could be used for reactive navigation. The focus will be made in what follows on the case of global path/trajectory planning to perform cognitive navigation (from the initial robot configuration to a desired final configuration in the environment). APF methods [11,14] are among the most disseminated works in the literature due mainly to their simplicity and intuitive use. For instance, authors in [18] used this technique for planning and re-planning missions for a group of mobile robots in cluttered and dynamic environment. Although the obtained results gave an interesting tool for decision making (which trajectory to take by each robot) in complex context as the multi-robot and multi-target reaching, the obtained trajectories did not take into account neither the robots' structural constraints nor the effective obstacles shapes. Therefore, in certain configurations the planned mission cannot be achieved. It is to be noted also that the main drawbacks of the APF techniques are linked to the oscillations and local-minimum configurations [19]. Several other techniques, use explicitly the obstacles shapes/dimensions to obtain the suitable robot's path. For instance, Voronoï diagrams [12] permits to the robot to navigate at equidistant value from all the surrounded obstacles. The obtained path corresponds therefore to the safest path, but it does not integrate, with straightway, neither other optimal criteria nor the robot's kinematics constraints. A large class of road-map based techniques [15,16] uses optimization process to choose one trajectory among a set of admissible ones [20–22]. For instance, optimal RRT (symbolized by RRT*) [22] permits to obtain a random tree containing an important number of admissible trajectories which could take into account the robot kinematics constraints. Once the overall tree is obtained, the algorithm consists to choose the optimal trajectory according mainly to a safety criteria and/or to the minimum path length. In RRT as well as in RRT*, each branch of the tree is expanded in privileged direction given by a probabilistic process [15,22]. The new branches are obtained usually using simple maneuvers, while maintaining the robot velocities constant (linear and angular) for a certain amount of time δT. According to the number of iterations but notably also to the number of the chosen discrete velocities values and to δT, the C_{Free} environment could be explored exhaustively or not. In fact, more the number of discrete possible velocities is high and δT small, more is important the explored C_{Free} and consequently also the time to obtain the result. In the above cited methods, it is possible to deal with a changing environments while regularly re-planning the robot's trajectory [23,24]. Therefore, in real motion conditions where the environment could be very cluttered, uncertain and/or highly dynamic, these methods may not be efficient, due among others things, to significant computational time related to these planning and re-planning phases [7,22,25].

The other part of the literature, categorized as reactive, considers that a robot needs to respond in real time to its current perceptions [26–28] and requires therefore less knowledge about the overall environment state. Local sensor information is used rather than a prior important knowledge on the environment [29–31]. The robot reacts therefore with stimuli-response behavior (generally qualified as bio-inspired [32]) and it does not need any important planning process to achieve the navigation. In fully reactive navigation, at each sample time, the robot must follow a defined set-points, according to its local perceptions and current objectives (for instance, reach a pre-defined location). The obstacles/walls/pedestrians/etc. are therefore discovered and avoided in real time. In [11] the author proposes a real time obstacle avoidance approach based on the principle of artificial potential fields. He assumes that the robot's actions are guided by the sum of attractive and repulsive fields. In [27] author extends Khatib's approach while proposing specific schema motors for mobile robot navigation. Another interesting approach, based on a reflex behavior reaction, uses the Deformable Virtual Zone (DVZ) concept, in which the robot's movements depend on risk zone surrounding the robot [33]. If an obstacle is detected, it will deform its DVZ and the approach consists of minimizing this deformation by modifying the control vector. This method deals with any obstacle shape, however, it suffers as schema motors from local minima problems. In general, this school of thought (the reactive one) does not require high computational complexity since the robot's actions must be given in real time according to local perceptions [34]. Obviously while taking this approach for robot's navigation, the overall robot movement (from its current robot location to the final desired location) cannot be considered as optimal, mainly if the environment is complex with a multitude of obstacles and maybe trapped regions [35]. In fact, since the robot's actions are guided only by local perceptions, the global optimality cannot be demonstrated in advance.

1.2. Hybrid control architecture

More and more control architectures exhibiting hybrid structure (reactive/cognitive) appear in the literature. This allows to take the advantages of the both structures while minimizing their drawbacks [36–40]. An interesting survey of 22 control architectures has been given in [39], highlighting among other things, how developments for Unmanned Grounded Vehicles (UGVs) architectures have been extended for autonomous underwater vehicles (the addition of the 3D navigation and the specificity of the sea environment). Usually in the literature, a consensus is adopted for the structure of hybrid control architecture which is generally structured in three layers: the highest level is responsible for mission planning and re-planning; the intermediate layer activates the low-level behaviors and permits passing of parameters to them; while the lowest level layer is the reactive layer and contains the physical sensor and actuator interfaces. The cognitive part (highest level) contains generally a symbolic world model (based on artificial intelligence concepts), which develops plans and makes decisions on the way to perform the robot's objectives. The reactive part (the two other lower levels) are responsible for reacting to local events without complex reasoning. Nevertheless, generally the structural conception of these hybrid architectures remain too complex to manage the different levels of hierarchy imposed by this kind of architectures [41–43]. They are also low harmonized to deal with the effective set-points to send to the robot actuators (lowest level). Efforts have concentrated on the conceptual aspects (using for instance the multi-agent paradigm to manage the multi-layered proposed architectures [44–46]) and less on the overall architecture simplicity, genericity and its effective implementation on the robot [47,18]. Indeed, even if the control architecture must show a good level of knowledge abstraction and decision, it is important to translate them in terms of low-level robot control set-points to exhibit clearly its effects on the robot's movements, which permit at its turn to attest on the safety and on the overall control architecture stability [31,48].

This paper proposes a new Homogeneous and Hybrid (reactive/cognitive) Control Architecture (HHCA) for vehicles navigating in different kinds of environments (cf. Section 4). This architecture permits to simply link the set-points defined by the cognitive or reactive levels to the effective control of the vehicle's movements. It is to be noted that the proposed hybrid control architecture uses a new proposed flexible and safe limit-cycle trajectories, called Parallel Elliptic Limit-Cycle (PELC, cf. Section 3.1.1) to perform either reactive (cf. Section 4.4.1) or cognitive (cf. Section 4.4.2) vehicle's navigation. The proposed PELC, could be used either as

(instantaneous or short term) a local planner or as a global planner (more cognitive approach) when the overall environment is well known. The proposed PELC constitutes therefore a homogeneous and accurate tool to define the vehicle's behaviors and missions. It is important to mention also that either in cognitive or reactive mode, the vehicle is controlled with the same control law (cf. Section 4.2). The proposed overall control architecture takes into account: robot structural constraints (e.g., nonholonomy); avoid command discontinuities and set-points jerking, etc. Thus, the efficiency, the safety and the stability of the overall control architecture could be rigorously demonstrated and analyzed. Otherwise, an appropriate hierarchical process of selection is defined to manage the different navigation contexts (cf. Section 4.4.3).

The rest of the paper is organized as follows. Section 2 presents the adopted overall navigation framework and the specification of the task to achieve. In Section 3 the different proposed elementary components for planning will be detailed before to focus on the multi-criteria optimization to perform optimal local and global path planning. Section 4 gives the details and the specificities of the proposed hybrid control architecture while presenting its different constituting modules. Section 5 is devoted to the description and the analysis of an extensive simulation results and first experiments. This paper ends with some conclusions and further work.

2. Overall navigation framework definition

Autonomous vehicle navigation aims, in the proposed generic framework, to lead the vehicle from its initial configuration, to a final configuration (called final target) while avoiding any obstacle (which could have different shapes, cf. Fig. 1). This navigation could be done even with reactive control (while acting online according to the vehicle's local perception, cf. Section 4.4.1) or with cognitive control (while following an already planned trajectory, cf. Section 4.4.2). The desired vehicle's movement needs to be safe and smooth along all its displacement. One supposes in the setup that the vehicle and the final target to reach are surrounded by circle shapes with a radius R_R and R_T respectively (cf. Fig. 1). For the obstacles/walls, it is supposed that they can be surrounded by appropriates ellipses (cf. Fig. 1), given by Eq. (1). An ellipse is defined in what follows by:

$$a(x-h)^2 + b(y-k)^2 + c(x-h)(y-k) = 1 \quad (1)$$

where:

- $h, k \in \mathbb{R}$, are the coordinates of the ellipse center,
- $a \in \mathbb{R}^+$, is related to the half-length $A = 1/\sqrt{a}$ of the ellipse's longer side (major axis),
- $b \in \mathbb{R}^+$, is related to the half-length $B = 1/\sqrt{b}$ of the ellipse's shorter side (minor axis), thus $b \geq a$,
- $c \in \mathbb{R}$, is related to the ellipse orientation (if $a \neq b$) $\Omega = 0.5atan(c/(b-a))$ (cf. Fig. 1). When $a = b$ Eq. (1) becomes a circle equation (Ω will not give thus any more information).

The Surrounded Ellipse's (SE) parameters (h, k, A, B and Ω) (cf. Eq. (1) and Fig. 1) can be obtained by the mobile robot either offline (using for instance a road map of the static environment) or online using for example a camera positioned in the environment [49] or the robot's telemetric sensors [50]. Among the challenging aspects when the robot navigates in fully reactive way (thus with discovering online its environment, cf. Section 4.4.1), is to update smoothly and efficiently the ellipses' parameters as the robot discovers the entire shape of the obstacles. In order to perform this important perceptive functionality, an appropriate weighted least square method, on the range data given by telemetric sensors, has been used in [51]. An extension of this last approach while using Extended Kalman Filter and an appropriate sub-optimal heuristic method has been developed in [50] and [52].

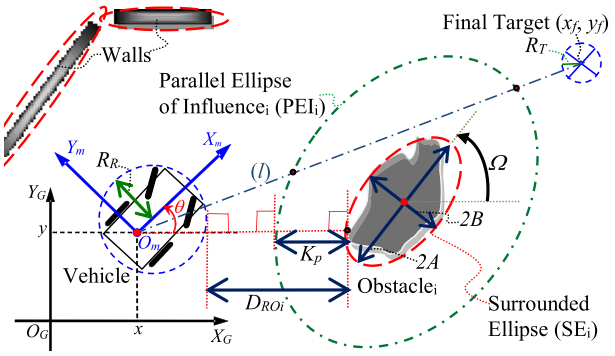


Fig. 1. Vehicle's pose and its perceptions for mainly a reactive navigation (cf. Section 4.4.1). The straight line "l" and the distance D_{RO_i} are used in Algorithm 3 to obtain the most obstructing obstacle. The parameters (A, B and Ω) characterize the SE$_i$ and the PEI$_i$ has a constant offset K_p w.r.t. SE$_i$ (cf. Section 3.1.1).

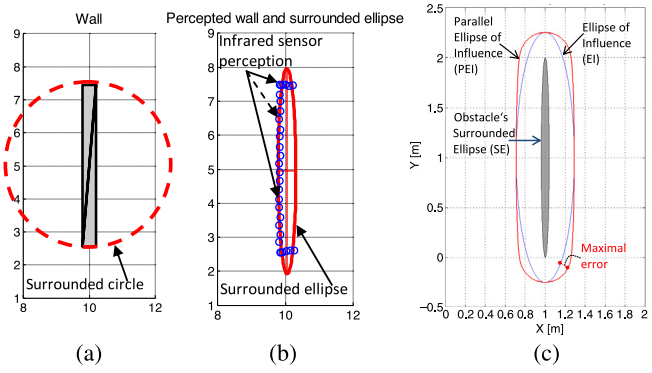


Fig. 2. Different shapes to surround and to safely avoid obstacles. (a) and (b) Interpolated wall using, respectively, a circle and an ellipse shape. (c) Difference between Parallel Ellipse and Ellipse of Influence (cf. Section 3.1.1).

It is to be noted that the choice to surround obstacles by an ellipse shape rather than a circle, as used in several works (as given in [53,54] or [55]), is motivated by the fact having one more generic and flexible component to surround and fit accurately different obstacles shapes. Among the examples of shapes which can be appropriately fitted with an ellipse and less by a circle is a wall (or in general, any longitudinal or thin objects). Fig. 2 shows this kind of configuration. In fact, if we would like to surround the wall given in this figure by an appropriate circle, this one will have a large radius which induces longer path to avoid it safely [53] (cf. Fig. 2(a)). Fig. 2(b) shows that the ellipse better fits the shape of this wall. This figure shows also uncertain perceptions obtained by infrared sensors on one side of the wall (left side). Several examples of vehicle navigation in cluttered and urban/structured environments will be shown in Section 5. These simulations will highlight, among other things, the validity and the relevance to surround different obstacles/walls/sidewalk/etc. by appropriate ellipses shapes.

After introducing the overall navigation framework and the different conventions, let us give the details of the main proposed components to achieve autonomous vehicle navigation.

3. Reliable path planning based on Parallel Elliptic Limit-Cycle (PELC)

Before giving the details about the proposed optimal local and global path planning (cf. Section 3.2), let us introduce first in Section 3.1.1 the common used elementary safe path (for reactive and/or cognitive navigation), based on a new generic mathematical

formulation of limit-cycles. In Section 3.1.2, an appropriate reference frame is proposed to define the vehicle suitable behavior (e.g., attraction, repulsion, avoidance, following).

3.1. Generic components

3.1.1. Elementary PELC

The navigation methodologies based on limit-cycles have been used in the literature to perform mainly intuitive and efficient obstacle avoidance behavior [53–56]. They are defined according to a circular [55] or an elliptic [57] periodic orbit (in the case of an elliptic orbit, this orbit corresponds to an Ellipse of Influence (EI), as given in Fig. 2). These periodic orbits can guarantee, if they are well-dimensioned (far enough from any obstacle) and accurately followed, to avoid any obstructing obstacle. The modeling of Limit-cycles is close to Dynamical System approach (DS, by dynamical systems it is meant a coupled set of "n" autonomous first-order differential equations) [58]. This well-known methodology allows while following its trajectories (DS set-points) to safely avoid obstacles by bypassing them. Nevertheless, DS requires very complex mathematical modeling to avoid any obstacle's shape. Contrary to that, the defined PELC does not need any complex computation, from the moment that the parameters of the surrounded ellipse are obtained ((h, k), A, B, Ω, cf. Eq. (1) and Fig. 1). In addition to obstacle avoidance, the proposed PELC can be easily used for several vehicle navigation sub-tasks (cf. Section 4.3). Therefore, it constitutes a uniform tool to perform safe and reliable navigation. The work given in [57] has permitted us to extend possible circular orbital limit-cycles (COLC) to elliptical orbital limit-cycles (EOLC, represented by PEI in Fig. 1), where COLC is only a particular case of EOLC when the major axis is equal to the minor axis. In this paper, an even more generic formulation of limit-cycle trajectories is proposed, permitting us to obtain an orbital Parallel Elliptic Limit-Cycle (PELC). This PELC has a constant offset according to the surrounded ellipse (cf. Fig. 1). Before giving more details about the new proposed PELC, let us give a short definition of the general principle of parallel curves, frequently called for mathematical definition "offset curves" [59]. These curves are obtained from a base curve by a constant offset, either positive or negative, in the direction of the curve's normal. The two branches of the parallel curve at a distance K_p away from a parametric representation of the base curve $(f(t), g(t))$ are given by:

$$x = f \pm \frac{K_p \dot{g}}{\sqrt{\dot{f}^2 + \dot{g}^2}}; y = g \mp \frac{K_p \dot{f}}{\sqrt{\dot{f}^2 + \dot{g}^2}}; \quad (2)$$

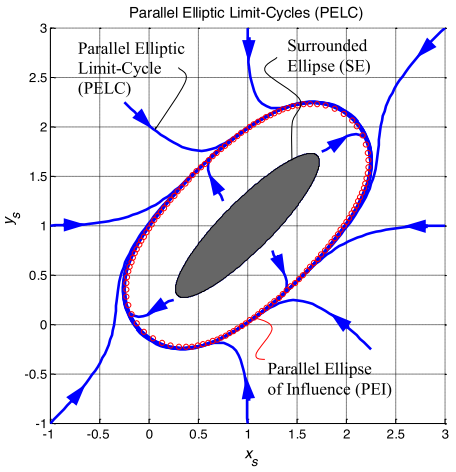
where $\dot{f} = df/dt$ and $\dot{g} = dg/dt$. It is important to mention that a parallel curve to an Ellipse is not an Ellipse (but an equation of 8^{th} order) [60], except obviously if the offset $K_p = 0$.

The differential equations defining the proposed PELC are given as follows (cf. Fig. 3(a)):

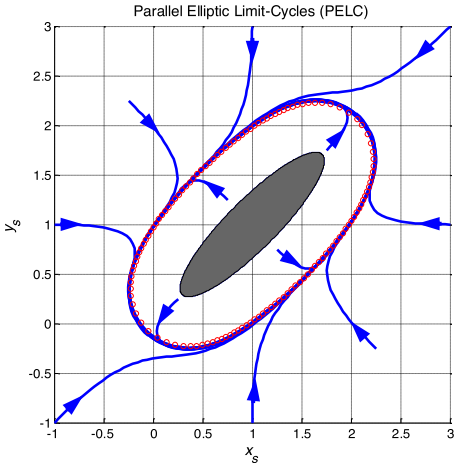
$$\begin{aligned} \dot{x}_s &= r y_s + \mu x_s (1 - \Psi) \\ \dot{y}_s &= -r x_s + \mu y_s (1 - \Psi) \end{aligned} \quad (3)$$

where:

- $\Psi = [4(z_1^2 + 3z_2)(z_2^2 + 3z_1 z_3) - (z_1 z_2)^2 + 18 z_1 z_2 z_3]/(9z_3)^2$, with: $z_1 = x_s^2 + y_s^2 - K_p^2 - A^2 - B^2$; $z_2 = B^2 x_s^2 + A^2 y_s^2 - A^2 K_p^2 - B^2 K_p^2 - A^2 B^2$ and $z_3 = (ABK_p)^2$.
- (x_s, y_s) correspond to the position of the vehicle according to the Surrounded Ellipse (SE) center (cf. Fig. 1).
- $r = 1$ for clockwise trajectories (cf. Fig. 3(a)) and $r = -1$ for counter-clockwise trajectories (cf. Fig. 3(b)).
- A and B characterize, respectively, major and minor SE axes (cf. Fig. 1).



(a) Clockwise.



(b) Counter-clockwise.

Fig. 3. Shape possibilities for the used Parallel Elliptic Limit-Cycles.

- K_p corresponds to the offset of the PELC with regards to SE. This offset is equal generally to the dimension of the vehicle (R_R) (cf. Fig. 1) plus a certain *margin* which corresponds to a safety tolerance including: perception uncertainty, control accuracy, etc.
- $\mu \in \mathbb{R}^+$ a positive constant value which allows us to modulate the convergence of the PELC. The convergence is as slow as μ is smaller, which permits us also to obtain smoother PELC (cf. Fig. 4).

The PELC given by Eq. (3) could also be defined according to an absolute reference frame (cf. Fig. 1). Indeed, it is enough to apply a translation (h, k) and a rotation Ω w.r.t. this absolute reference frame. In Fig. 3 (a) and (b), the shown PELCs are characterized by: a center (1, 1), i.e., $h = 1$ and $k = 1$; an orientation $\Omega = \pi/4$; a half major and minor axes equal respectively to $A = 1$ and $B = 0.25$; an offset $K_p = 0.5$, and finally $\mu = 1$. It is observed that the PEI given by $(h, k, \Omega, A, B, K_p) = (1, 1, \pi/4, 1, 0.25, 0.5)$ is a periodic orbit.

The trajectory which has this kind of periodic orbit is called a Parallel Elliptic Limit-Cycle (PELC). The trajectories from all points of the $X_G Y_G$ global reference frame, including inside the PEI, move toward it (cf. Fig. 3). To demonstrate analytically the validity of the result for any initial state of the PELC, the following Lyapunov function is used:

$$V(x) = (1/2)(x_s^2 + y_s^2). \quad (4)$$

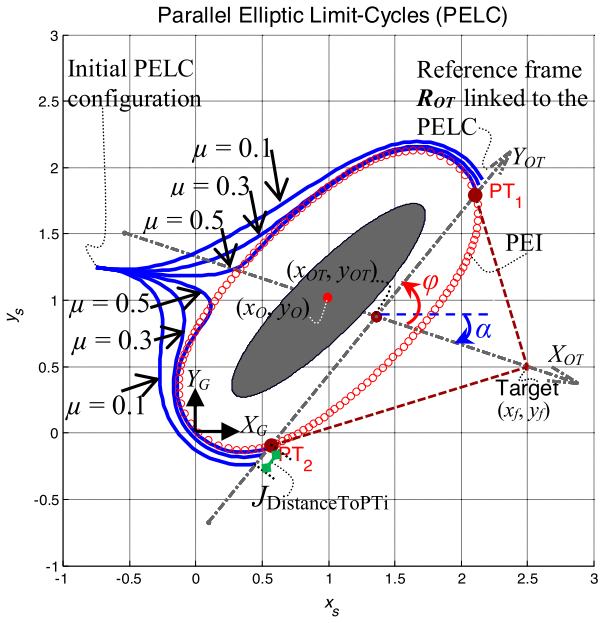


Fig. 4. Reference frame linked to each PELC and different PELC shapes according to the value of μ. $J_{\text{DistanceToPTi}}$ will be used in Section 3.2.1 as a sub-criteria to evaluate each PELC.

$V(x)$ gives thus an indication of the distance progress between the point (x_s, y_s) and the origin $(0, 0)$. The derivative of $V(x)$ along the trajectories of the system is given by:

$$\dot{V}(x) = x_s\dot{x}_s + y_s\dot{y}_s.$$

After replacing $\dot{x}_s$ and $\dot{y}_s$ with equations given in (3), it is obtained finally:

$$\dot{V}(x) = \mu V(x)(1 - \Psi). \tag{5}$$

Knowing that $\mu V(x)$ is always positive, the sign of the derivative of $\dot{V}(x)$ is defined:

- Negative if $(1 - \Psi) < 0$, thus if the initial condition (x_{s0}, y_{s0}) is outside the PEI (given by equation $\Psi = 1$), which means that $V(x)$ will decrease since $(1 - \Psi) < 0$,
- Positive if the initial condition is inside the PEI $(1 - \Psi) > 0$, which means that $V(x)$ will increase.

Therefore, according to this two ensured behaviors of $V(x)$ (negative or positive progress) according where the point is (outside or inside the PEI given by $\Psi = 1$), allows us proving that $\Psi = 1$ is always a periodic orbit of the PELC. One can also say from Eq. (5) that, higher is the value of μ, more is the limit-cycle velocity converges toward its periodic orbit (the opposite is true, cf. Fig. 4).

It is important to notice that contrary to a standard Ellipse of Influence (EI, as defined in [57] (cf. Fig. 2(c))) the PEI (the PELC final orbit attractor) allows to always guarantee an effective minimal distance w.r.t. the contours of the ellipse surrounding the obstacle (SE in Fig. 2(c)). This constant offset is equal to K_p as given above. Indeed, while using the formulations given in [57], the EI is obtained while fixing its half major A_{lc} and minor B_{lc} axes according to (A, B) of the SE, it was used $A_{lc} = A + K_p$ and $B_{lc} = B + K_p$. Nevertheless, this formulation permits us to ensure accurately this minimal distance (K_p) only in 4 points corresponding to minor and major end-point axes, as given in Fig. 2(c). Indeed, in this figure it is shown the difference between obtained PEI and EI, when: $A = 1m$; $B = 0.05m$ and $K_p = 0.25m$. As maximum difference, a maximal error of $0.1m$ is obtained, which corresponds to an error rate of 40%, which is, too important if we want to ensure safe vehicle navigation in all situations. Obviously we can increase, according to this maximum error rate, the value of K_p, but this will not lead to an optimal path length. It is also important to note that the error rate is as big as the obstacle is more longitudinal (i.e., $A \gg B$). The proposed PELC allows, in addition to having safer and reliable navigation, to obtain smooth and flexible navigation in different environments (e.g., cluttered or not, structured or not, cf. Section 5). For instance, the walls/sidewalks limits in an urban environment could be surrounded by a very thin PEI (cf. Section 5.1.1).

Once the formulation and the interesting features of the proposed PELC have been given, let us introduce in the following Section 3.1.2 the details of their use for obstacle avoidance as well as for simple attractors toward specific targets in the environment. These PELC trajectories will constitute therefore a common tool for different vehicle's behaviors, and it will be used either for reactive or cognitive navigation (cf. Section 4.4).

3.1.2. References frame linked to the task achievement

For simple and efficient description of vehicle navigation in any kind of environment, it is proposed in what follows a specific reference frame assigned for each obstacle / wall / target / etc. inside the considered environment (or at least for each element inside the vehicle's field of view). These specific references frames will guide the vehicle behaviors and permit us to evaluate the success of the current achieved sub-task (e.g., wall following, obstacle avoidance, target reaching/tracking, etc.). Each elementary reference frame will orient thus locally the achievement of the vehicle navigation toward its final objective. A kind of analogy could be established with robot manipulator modeling. In fact, when we would like to control the movement of a robot end-effector (w.r.t. its base), it is assigned for each articulation an appropriate reference frame while using dedicated conventions [61,62]. These local reference frames are mainly used to express simply the local elementary articulations' movements (translation / rotations) in order to obtain the desired final end-effector movement. The context of vehicle navigation is obviously different but the proposed reference frames will similarly lead to make a reasoning on the efficiency of the vehicle movements in order to reach its final objective. To define each specific reference frame, it is mandatory to fix its center and its axes orientation. These will be specified in what follows.

Let us start to define the reference frame linked to each obstacle (wall or any object which could obstruct the vehicle's movement). These one will have a specific reference frame $(\Re_{OT})$ permitting us to define the obstacle avoidance sub-task achievement while knowing the localization of the vehicle according to it (cf. Section 4.4). $\Re_{OT}$ is obtained with a simple geometric construction and has the following features (cf. Fig. 4):

- X_{OT} axis connects the center of the obstacle (x_O, y_O) to the center of the final Target (x_f, y_f). This axis is oriented toward this target.
- the Y_{OT} axis is defined by two points PT_1 and PT_2, which correspond to the tangent points between the two straight lines coming from the final target (x_f, y_f) and the Parallel Ellipse of Influence (PEI). Y_{OT} axis is oriented while following trigonometric convention.

The axes X_{OT} and Y_{OT} intersect on the point $O_{OT} = (x_{OT}, y_{OT})$ and they have an angle φ between them. To guide the vehicle's future movements (cf. Section 4.4), it is important to define its localization w.r.t. $\Re_{OT}$. One needs, therefore, to make a transformation from the global reference frame $X_G Y_G$ to the local reference frame $X_{OT} Y_{OT}$. Knowing that $X_{OT} Y_{OT}$ is not necessarily orthogonal, it proceeds with two steps:

1. Transformation of the vehicle localization $(x, y)_G$ from the global reference frame to an Intermediate orthonormal reference frame which has as center (x_{OT}, y_{OT}) and X axis = X_{OT}. The following homogeneous transformation is used (cf. Fig. 4):

$$\begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix}_{OT_I} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & x_{OT} \\ \sin\alpha & \cos\alpha & 0 & y_{OT} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix}_G . \quad (6)$$

2. Once x_{OT_I} and y_{OT_I} are obtained for this Intermediate transformation, we can obtain finally the localization of the vehicle (x_{RO}, y_{RO}) w.r.t. $\Re_{OT}$ as follows (cf. Fig. 4):

$$\begin{cases} x_{RO} = x_{OT_I} - y_{OT_I} \dfrac{\cos(\varphi - \alpha)}{\sin(\varphi - \alpha)} \\ y_{RO} = \dfrac{y_{OT_I}}{\sin(\varphi - \alpha)}. \end{cases} \quad (7)$$

It is to be mentioned that $(\varphi - \alpha) \neq 0$ modulo π. It is always true because the axes X_{OT} and Y_{OT} are never parallel (or co-linear) from the moment that: (x_f, y_f) is outside of the PEI attributed to the considered obstacle and that the obstacle exists effectively (its A and B features $\neq 0$). According to that, $\sin(\varphi - \alpha)$ (used in Eq. (7)) is always $\neq 0$.

The main idea to use this essential component $(\Re_{OT})$ is to determine which PELC parameters, the vehicle must follow to avoid for example an obstacle. In fact, once the transformation from the global frame $X_G Y_G$ to the local reference frame $X_{OT} Y_{OT}$ is obtained, it is enough for instance to check the sign of the vehicle's localization according to the axis X_{OT} to assign the vehicle appropriate behavior (cf. Section 4.4). For instance, if the sign of x_{RO} is negative, the vehicle must follows the defined PELC (to avoid the obstacle) and if positive the vehicle can consider that the obstacle is not an obstructing obstacle and can go thus straight toward its final target (cf. Fig. 4). At the condition obviously that there is no other constrained obstacle; if not, the process will be reiterated (cf. Section 4.4.1).

In previous works [55,57], the sign y_{RO} (ordinate of the robot in $\Re_{OT}$) has been used to determine the right direction to avoid the obstacle. If $y_{RO} \geq 0$ then apply clockwise limit-cycle direction else apply counter-clockwise direction (cf. Fig. 3). This simple rule permits to reduce the length of robot trajectory in the case where the obstacle is surrounded by a circle. In Section 3.2.1, the optimality of the followed PELC will be presented while determining its best direction and shape. This will be done while obtaining the optimal value of μ (cf. Eq. (3)).

It is important to note that each final target $T = (x_f, y_f)$ in the environment will be assigned also a reference frame $(\Re_T)$ which has as a center (x_f, y_f) and as axes, orthogonal one, oriented as the axes of the global reference frame. This description permits the homogenization of using these references frames for all the objects/targets present in the environment. The pertinence of these references frames will be shown notably in the following section.

3.2. Optimal local and global path planning

It is proposed in what follows to obtain a generic way to enhance the use of the already presented PELC, to perform optimal: local obstacle avoidance as well as global robot navigation in cluttered environments. These optimized components (PELC* (cf. Section 3.2.1) and gPELC* (cf. Section 3.2.2)) will be afterward integrated in a Hybrid (reactive/cognitive) multi-controller architecture (cf. Section 4) and will constitute a homogeneous way to obtain the vehicle's set-points. It is to be noted that further discussions about the proposed multi-criteria optimization will be given in Section 3.2.3.

3.2.1. Local path generation based on PELC*

Knowing the features of the Surrounded Ellipse (cf. Fig. 1) (i.e., h, k, Ω, A and B) and the desired offset (safe distance K_p (cf. Eq. (3))) to the obstacle, it is shown in what follows how to obtain the optimal value of μ^* which permits us to minimize a multi-criteria function. This latter, called J (cf. Eq. (8)), gather different important sub-criteria linked to the features of the corresponding PELC path. Indeed, according to the value of μ in the PELC Eq. (3), the shape of the obtained limit-cycle can converge quickly or not to the assigned PEI (cf. Fig. 4), but what is the optimal value of μ^* to minimize the multi-criteria J? It is important to note that J evaluates the obtained PELC path while taking into account: the vehicle kinematic constrains, the actual applied control law (cf. Section 4.2), the initial vehicle configuration (x, y, θ) (cf. Fig. 1) and its final reached configuration (corresponding in what follows to the PELC configuration when it reaches the axis Y_{OT} (obstacle) or Y_T (final target) (corresponding to $J_{\text{DistanceToPTi}}$ and $J_{\text{Distance\_gPELC\_FinalTarget}}$ in Fig. 4 and Eq. (12) respectively)). These considerations permit us to obtain new planned paths based on PELC which can be actually followed by the vehicle.

$$J = w_1 J_{\text{DistanceToPTi}} + (1 - w_1) J_{\text{PELC}_{\text{Length}}} + w_2 J_{\text{PELC}_{\text{Curvature}}} + w_3 Bool_{\text{MaximumCurvature}} + w_4 Bool_{\text{Collision}} \quad (8)$$

where:

- $w_i \mid i = 1..4 \in \mathbb{R}^+$ are constants permitting to give the right balance between the different sub-criteria characterizing the computed PELC. The criteria J is defined therefore according to the vector $W_{PELC} = \{w_1, w_2, w_3, w_4\}$. It is to be noted that the weight $w_1 \in [0\ 1]$ and permits, as given in Eq. (8), to balance between the values of $J_{\text{DistanceToPTi}}$ and $J_{\text{PELC}_{\text{Length}}}$ sub-criteria.
- $J_{\text{DistanceToPTi}}$ corresponds to the distance between the final reached position of the computed PELC and one among the points $\text{PT}_i \mid i = 1..2$ (chosen according to whether the PELC is obtained for clockwise or counter-clockwise avoidance (cf. "r" in Eq. (3)). For instance, Fig. 4 shows the value of $J_{\text{DistanceToPTi=2}}$ for a counter-clockwise PELC with $\mu = 0.1$.
- $J_{\text{PELC}_{\text{Length}}}$ corresponds to the curvilinear length of the obtained PELC. It is computed using the following equation.

$$J_{\text{PELC}_{\text{Length}}} = \int_{s_0}^{s_f} ds \quad (9)$$

  where s_0 and s_f correspond respectively to initial and final curvilinear abscissa of the obtained PELC.
- $J_{\text{PELC}_{\text{Curvature}}}$ characterizes the PELC curvature along its length. It is computed using the following equation.

$$J_{\text{PELC}_{\text{Curvature}}} = \int_{s_0}^{s_f} \mathcal{C}(s)^2 \, ds \quad (10)$$

  where $\mathcal{C}(s)$ is the curvature at the abscissa s, $\mathcal{C}(s) = 1/\rho(s)$ with $\rho(s)$ the radius of curvature at s abscissa.
- $Bool_{\text{MaximumCurvature}}$ corresponds to a Boolean value, which is equal to 1 if the maximum possible vehicle curvature is reached (cf. Section 4.2). It is to be noted that the weight w_3, linked to this sub-criteria, is a big positive value so that the overall criteria J will be highly penalized if the obtained PELC has at least one configuration where the vehicle must attain its maximum curvature (maximum steering angle) [63].
- $Bool_{\text{Collision}}$ corresponds to a Boolean value, which is equal to 1 if the vehicle collides with at least one Surrounded Ellipse in the environment (cf. Fig. 1). It is to be noted that the weight w_4, linked to this sub-criteria, is a very big positive value $\longrightarrow \infty$ so that the overall criteria $J \longrightarrow \infty$ if any obstacle is collided.

To obtain μ^* optimizing the PELC (cf. Eq. (3)) according to the multi-objective function J, the parametric optimization $\partial J/\partial \mu = 0$ should be computed. The mathematical formulation of this problem is highly non-linear, and the analytic solution is therefore complex to get. In this paper a numerical optimization (using iterative dichotomy for instance) is used to obtain μ^*.

An important assumption in the proposed optimization is the fact that the vehicle, in order to perform its navigation, must deal with sequentially only one obstacle/target at a time, until reaching its corresponding axis Y_{OT} (or Y_T), and switching to the other obstacle/target and so on until reaching the Y_T axis of the final target. In each elementary optimization, the obtained PELC* will permit either an obstacle avoidance behavior or an attraction toward a target. In these two optimizations, the parameters of the obtained PELC* are respectively:

- PELC*$((h, k), \Omega, (A, B, K_p), \mu^*)$, where h, k, Ω, A and B are the features of the detected obstacle and K_p the desired safe distance to this obstacle.
- PELC*$((x_f, y_f), 0, (\xi, \xi, \xi), \mu^*)$, where (x_f, y_f) is the position of the final target to reach and ξ is a very small value $\longrightarrow 0$.

This common description based on PELC permits us to have a homogeneous way to perform all the possible vehicle behaviors (i.e., obstacle avoidance and target reaching). Indeed, since in all the cases, the PELC will converge toward its PEI, it is enough to tell that even for the target reaching behavior, a PELC could be used efficiently. It is to be noted, as given in Section 2 and in Fig. 1, the main target to reach has a radius R_T, which means that the target is considered reached as soon as the robot-target distance is lower than R_T.

These two elementary behaviors will be used to deal with different environments (cf. Sections 4.4 and 5.1) while performing either reactive or cognitive navigation, but first, let us introduce the proposed methodology to obtain long-term vehicle planning while using a sequence of appropriate PELC.

3.2.2. Global path generation based on gPELC*

This subsection has as objective to present the proposed optimal overall methodology, based on PELC, to lead the vehicle from its initial posture $P_0 = (x_0, y_0, \theta_0, \gamma_0)$ (cf. the used vehicle model in Section 4.2) to a final assigned posture $P_f = (x_f, y_f, \Xi, \Xi)$, where (x_f, y_f) corresponds to the position of the final target and Ξ symbol means here, any real value. Indeed, in this paper, the values of the final vehicle's heading θ_f and front wheels angle γ_f are not imposed. The optimal methodology aims to connect several PELC to reach P_f while allowing us to guarantee the safety and the smoothness of the obtained global path-based on PELC (called gPELC). The targeted smooth path will permit to generate smooth set-points for the control law, allowing thus to avoid the actuators jerking which ensures the passengers' comfort and preserves the actuators' lifetime [6]. Obviously, the aim of the proposed optimal methodology is to ensure also the continuity of the vehicle heading θ and the wheels orientation γ (cf. Fig. 7), even at the connection point between the PELCs.

To formalize the optimal path planning using a sequential concatenation of PELCs (gPELC*), let us use Graph Theory [64,65], through a shortest-path problem to optimize the several possible gPELCs. In graph theory, the shortest-path problem corresponds to finding a path between any two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. It is shown in Section 3.2.1 that according to the value of μ the shape of the obtained PELC changes (cf. Fig. 4) and consequentially the value of J (cf. Eq. (8)). The main idea is therefore to obtain the sequence of elementary PELC (with appropriate values of μ and direction (clockwise or not, according to the value of r in Eq. (3)) which permit us to minimize the sum of costs leading the vehicle from P_0 to P_f.

It is supposed in what follows the presence of N obstacles in the environment, each one has an identification number id and is surrounded by an appropriate Parallel Ellipse of Influence (PEI_{id}, cf. Fig. 1). PEI_{id} is characterized by $[(h_{id}, k_{id}), \Omega_{id}, (A_{id}, B_{id}, K_{p\,id})]$ (cf. Eq. (1) and Section 3.1.1). id corresponds to the identifier of the obstacle $id = \{1, \ldots, N\}$ or to $id = f$ if the PEI is linked to the final target. For easy understanding of the proposed overall optimal planning given in Algorithm 1, some definitions / conventions will be given below to formalize the optimization problem, using graph theory, more specifically while using a Tree structure:

- A Tree (T) is a directed rooted Graph in which any two vertices are connected by exactly one path (without closed loops (cycles)). The tree T is characterized by T = (V, E), where V and E are respectively the set of all vertices and all the edges of the obtained T.
- Each vertex $v_j \in$ V is characterized by a state $v_j \equiv [(x_j, y_j, \theta_j, \gamma_j), \text{Parent}(v_j))] = [P_j, v_i]$ (where P_j corresponds to the vehicle's set-point when it will reach the vertex v_j). The tree root v_0 does not have any parent, $v_0 \equiv [P_0, 0]$. The vertex v_0 alone corresponds to level 0 ($\textbf{Level}_0$) of T and is characterized by a set of vertices represented by $\mathcal{S}_0 = \{v_0\}$. $\mathcal{S}_i$ will correspond therefore to $\textbf{Level}_i$ of the tree and will contain all the children vertices generated from vertices of $\mathcal{S}_{i-1}$ set. Each vertex $v_j \neq v_0$:
  - holds, as given above, the value of its parent v_i in T. We can write thus v_i = Parent(v_j) and v_j = Child(v_i), v_i and v_j are adjacent vertices.
  - contains the final state of one PELC_i^j (when first reaching the Y axis of the reference frame $\infty\, v_j$). The symbol "∞" expresses the fact that the considered vertex v_j is defined w.r.t. the reference frame linked to one obstacle or to the final target (cf. Section 3.1.2). In that case, it is written: $v_j \infty \Re_{id}$.
- Each edge $e_i^j \in$ E is characterized by a state $e_i^j \equiv [\text{PELC}_i^j, J_i^j]$, where $\text{PELC}_i^j \equiv \text{PELC}_i^j(\text{PEI}_{id}, r, \mu)$ corresponds to a Parallel Elliptic Limit-Cycle linking the vertex v_i to $v_j \infty \Re_{id}$. This PELC_i^j has as the initial state, the posture defined in v_i and as final state, the posture defined in v_j. PELC_i^j is characterized also by r and μ which correspond respectively to the PELC direction and specific shape (cf. Section 3.1.1 and Fig. 4). PELC_i^j permits to reach v_j from v_i (cf. Eq. (3)). The edge e_i^j has a weight $J_i^j \equiv J_i^j(\text{PELC}_i^j)$ corresponding to the PELC_i^j cost given by Eq. (8).
- The number of children (or also growing branches/edges) from each vertex v_i is fixed in the proposed Algorithm 1 through the pre-fixed constant $m \in \mathbb{N}^+$. This algorithm proposes to generate m PELC in each direction (clockwise and counter-clockwise, i.e., r will be equal to ± 1 respectively in Eq. (3)). In each direction, several PELCs are generated for each μ value given in a predefined set $S_\mu = \{\mu_1, \ldots, \mu_m\}$. Each generated PELC will be defined either according to the final target or to the obstacle_{id} (cf. Algorithm 1). It is to be noted that if $m = 1$, the idea is to generate an optimal PELC* (cf. Section 3.2.1) in clockwise and another in counter-clockwise direction respectively. Furthermore, if $m > 1$, then the chosen fixed values of μ are those which show the large shape possibilities of the PELC_i^j (slow and quick convergence toward the PEI_{id} (cf. Fig. 4), where $v_j \infty \Re_{id}$). For example if the slow and quick convergence correspond respectively to μ_{min} and μ_{max}, then the m values of $\mu \in S_\mu$ will be given by $\{\mu_{min}, \mu_{min} + \delta\mu, \ldots, \mu_{min} + (m -$

$1)\delta\mu, \mu_{max}\}$, where $\delta\mu = ((\mu_{max} - \mu_{min})/m)$. Obviously, more is important the value of m closest is gPELC* to the optimal effective path (linking P_0 to P_f). It is to be noted that if T has $n+1$ vertices, therefore T will have n edges. The tree's size, given by |E|, corresponds to the number of edges. The number of vertices if each vertex generates $2m$ children is given by the following formula:

$$1 + 2m + (2m)^2 + \cdots + (2m)^h = \frac{(2m)^{h+1} - 1}{2m - 1} \tag{11}$$

where h corresponds to the greatest level in T (called also the height of the rooted tree).

- A **valid global path**, defined by gPELC$_n$, is a path which starts from v_0 and reaches the vertex v_n (linked to the reference frame attributed to the main target $\Re_f$, it is noted therefore $v_n \circledcirc \Re_f$) without any obstacle collision. The gPELC$_n$ is obtained from an oriented graph given by a sequence of vertices $(v, v_1, \ldots, v_n) \in V^n$ such that v_{i-1} is adjacent to v_i and $i \in \{0, 1, \ldots, n\}$. gPELC$_n$ is a path of length n from v_0 to v_n. It is to be noted that the indexes given to v_i are variables and are not related to any canonical labeling of the vertices but only to their position in the sequence.
- The optimal path gPELC* is a **valid global path** that over all possible gPELC$_n$ minimizes the function:

$$G = \sum_{i=1}^{n} J_{i-1}^{i} = w_1 G_1 + (1 - w_1) G_{1Bis} + w_2 G_2 + w_3 G_3 + w_4 G_4 + w_5 J_{\text{Distance\_gPELC\_FinalTarget}} \tag{12}$$

where:

  - $w_i \mid i = 1..4 \in \mathbb{R}^+$ are the constants defined in Eq. (8), permitting to give the right balance between the different sub-criteria characterizing each elementary computed PELC, to give form to a gPELC. $w_5 \in \mathbb{R}^+$ permits to give more interest to the gPELC$_n$ which has a closest final point (gPELC$_n(s_f)$ where s_f, the final curvilinear abscissa of gPELC$_n$) to the final target P_f. This last information is embedded in the sub-criteria $J_{\text{Distance\_gPELC\_FinalTarget}}$.
  - The global criteria G is defined therefore according to the vector $W_{gPELC} = \{w_1, w_2, w_3, w_4, w_5\}$ and the sum of the elementary sub-criteria (cf. Section 3.2.1):
    - $G_1 = \sum_{i=1}^{n} J_{(i-1)\text{DistanceToPTi}}^{i}$
    - $G_{1Bis} = \sum_{i=1}^{n} J_{(i-1)\text{PELC}_{\text{Length}}}^{i}$
    - $G_2 = \sum_{i=1}^{n} J_{(i-1)\text{PELC}_{\text{Curvature}}}^{i}$
    - $G_3 = \sum_{i=1}^{n} J_{(i-1)Bool_{\text{MaximumCurvature}}}^{i}$
    - $G_4 = \sum_{i=1}^{n} J_{(i-1)Bool_{\text{Collision}}}^{i}$

To summarize, the idea to obtain the optimal gPELC* (cf. Algorithm 1) is to get the optimal sequence of elementary PELC to reach the main target P_f. The proposed Algorithm 1 permits to obtain a tree T, containing vertices linked with PELC without collisions, and with each edge weight obtained while using Eq. (8). Each valid gPELC permits to start from the vertex v_0 to reach vertices $\circledcirc \Re_f$. The gPELC* is the one which minimizes G (cf. Eq. (12)). Finally, the gPELC* contains the optimal sequence of local PELC$_i^j$ (with their values r_j and μ_j). Generally, once the tree T is available, the optimal-path from the root v_0 to a vertex v_n ($v_n \circledcirc \Re_f$) can be obtained while using for instance, tree-search-based on Dijkstra's algorithm [66] or the well-known Bellman–Ford algorithm.

In order to better explain the main steps of Algorithm 1, an enough simple environment has been used to perform gPELC* (cf. Fig. 5). As shown in this figure, Algorithm 1 proceeds first with trying to reach the main final target starting from the initial vehicle configuration by using the different PELC(μ) with different

Algorithm 1: Overall proposed methodology to obtain gPELC*

Data: Initial vehicle state; Environment features: Obstacles$_{id=1,\ldots,N}$ and Final target position; $S_\mu = \{\mu_{k=1,\ldots,m}\}$ the set of possible μ values.
Result: gPELC* (optimal global Parallel Elliptic Limit-Cycle)

```
1  //Initialization
2  S_0 = {v_0}; //The set of the vertices at Level_0 of the tree T//
3  i = 0;
4  while Not all the vertices of S_i ⊚ ℜ_f do
5      i = i + 1;
6      //Compute all the PELC: starting from vertices given in S_{i-1}
       (which are not ⊚ with ℜ_f) and targeting ℜ_f//
7      PELC^f_{[j=1,...Card(S_{i-1})]}(PEI_f, r = ±1, S_μ);
8      forall the Obtained PELC^f_j do
9          if It does not collide with any obstacle then
10             //v_{f#} exists thus, where "f#" corresponds to the index
               number of the vertex ⊚ ℜ_f. This implies that a valid
               global path could be obtained while knowing all the
               antecedents vertices of v_{f#}//
11             S_i ← v_{f#}; //Add the vertex to the tree T//
12             e_j^{f#} ← J(PELC^f_j(PEI_f, r, μ_k)); //Compute the weight of
               the edge e_j^{f#} //
13         else
14             Obtain the id of the first obstructing obstacle (cf.
               Algorithm 3);
15             //Compute all the PELC: starting from the vertex v_j and
               finishing in ℜ_id//
16             PELC_j^{id_{rμ}}(PEI_id, r = ±1, S_μ); //Where the index number
               "id_{rμ}", is linked to the value of r and μ, corresponds to
               the vertex index ⊚ ℜ_id//
17             if It does not collide with any other obstacle then
18                 if Exhaustive Expanding Tree then
19                     //Add all valid vertices
20                     S_i ← v_{id_{rμ}};
21                     e_j^{id_{rμ}} ← J(PELC_j^{id_{rμ}}(PEI_id, r, μ_k));
22                 else
23                     //Add only the optimal vertex in each
24                     direction (clockwise and counter-clock.)//
25                     S_i ← v*;
26                     e_j ← J*;
27                 end
28             end
29         end
30     end
31 end
32 //Apply Dijkstra's algorithm [66] on the obtained final tree T//
33 gPELC* = DijkstraAlgorithm(T);
```

possible values of μ (taken from the set S_μ which contains only one value in this simple example) this corresponds to lines 6 and 7 in Algorithm 1. Since this PELC path (numbered (1) and plotted with "+" symbols to mean invalid PELC, cf. Fig. 5) collides with obstacle O2, this PELC is stopped and its termination (defined by a vertex/node) will not be subject to future extension. After that, two other PELC(μ) are computed (in clockwise and counter-clockwise) in order to avoid the first obstructing obstacle (cf. lines 14 to 16 in Algorithm 1). The first PELC (numbered (2)) is stopped because it collides with O1 and not more extension will be possible from this PELC (cf. line 17), and the second (numbered (3)) corresponds to a

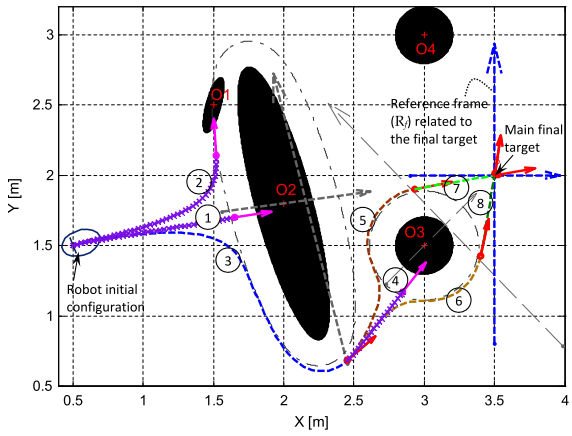


Fig. 5. Example of simple simulation highlighting the main steps of Algorithm 1. PELC paths given by "+" symbols mean invalid PELCs. The others elementary PELC in dotted colored lines show valid PELCs. The arrows given in solid lines correspond to the final PELCs configurations.

valid PELC and could be extended, and so on until reaching finally the reference frame related to the main target.

It is to be noted that the proposed planning based on gPELC* (cf. Algorithm 1) uses as RRT* [22] a tree of admissible paths (i.e., without collisions) while using elementary PELC, but contrary to RRT*, each individual path (branch) is, in addition to be safer, is also the most suitable (in certain cases, the optimal one (cf. Section 3.2.1)) to avoid each obstacle. The obtained overall path-based on gPELC* is therefore closer to the effective optimal path leading the robot toward its final destination. The computation time is also much smaller w.r.t. RRT*, mainly due to the reasoning proposed for the tree expansion (cf. Algorithm 1), whereas the tree in RRT* is only expanded (randomly) by a constant length, depending on the adopted constant robot's velocities and δT (cf. Section 1.1). Some other comments on Algorithm 1 are emphasized in what follows.

The **Else block** given between line 13 and 29 of Algorithm 1, expresses the fact that, when an obstacle$_{id}$ obstructs one extended PELC$_i^f$ then this obstacle$_{id}$ will be selected as an intermediate orbit (before reaching the Y axis of $\Re_f$). This will be done while computing another PELC$_i^{id}$, starting from the same vertex v_i but aiming to terminate in the Y axis of $\Re_{id}$ before adding a new vertex to T, permitting us to explore further this new branch. Nevertheless, if this new computed PELC$_i^{id}$ collides with any other obstacle (before reaching the Y axis of $\Re_{id}$), this branch is terminated without adding any new vertex to T. This choice is made to avoid certain infinite loops and to reduce the number of combination given by Algorithm 1. This supposition has been also made because if another collision is taken into account, it is obligatory to take this new obstacle *id* as a new intermediate orbit, which is in contradiction with the first supposition, consisting of addressing the case of the first obstacle$_{id}$ only because it is the first obstructing obstacle which does not allow PELC$_i^f$ to reach the Y axis of the main target $\Re_f$ (from the vertex v_i).

Inside this same **Else block** (line 13 to 29) there is also an important characteristic to highlight. It corresponds to the **If block** (between line 18 and 27) permitting to apply either an Exhaustive Expanding Tree (EET) or not. If yes, this corresponds to adding to the tree all the valid vertices and if no, then adding only the optimal vertices. Consequently, in the first case, the number of branches (PELC) from the vertex $v_{i-1} \circledcirc \Re_{i-1}$ to the reference frame $\circledcirc \Re_i$ could be at maximum equal to $2m$ (if all the computed PELC are valid) and in the second case, the maximum number of branches from v_{i-1} is 2 (which correspond respectively to the optimal PELC* for clockwise and counter-clockwise directions). The main objective of the second case is to reduce the number of possible expanding branches at each iteration of Algorithm 1. The aim is obviously to reduce the overall computation time to obtain the gPELC*. In this last version, the number of explored states is reduced but does not guarantee to have such a good solution gPELC* as given by EET. This result will be highlighted in the simulations given in Section 5.1. In that simulations, it is shown also that the solutions obtained with only optimal vertices expanding are generally not so far from the effective gPELC* (given by EET).

3.2.3. Multi-criteria optimality discussion

It is important to notice that as almost all multi-criteria optimization problem, there does not exist a single solution that simultaneously optimizes each sub-criteria [67,68]. The objective functions are said, in this situation, to be conflicting and there exists a (possibly infinite) number of Pareto optimal solutions. It is necessary therefore to define Pareto frontier with acceptable trade-offs between the different sub-criteria [67]. Precise analysis of the appropriate balance between each sub-criteria will be the subject of future work.

Nevertheless, according to the formalization of J (defined in Eq. (8)), it is important to highlight certain general tendencies: for instance, bigger is the value of μ bigger is the value of $J_{\text{PELC}_{\text{Curvature}}}$ (until reaching its maximum value) and lowest is the value $J_{\text{DistanceToPTi}}$ (until reaching its lowest value 0). In addition, it is important to say that $J_{\text{PELC}_{\text{Length}}}$ has a global minimum for a specific value of μ. The other two sub-criteria ($Bool_{\text{MaximumCurvature}}$ and $Bool_{\text{Collision}}$) are Boolean values obtained according respectively if the computed PELC reaches the maximum possible vehicle turning circle or if this PELC passes through at least one Surrounded Ellipse in the environment (cf. Section 3.2.1). Hence, according to the defined multi-criteria optimization problem, the general tendency of each sub-criteria, and also to the fixed priority between the different sub-criteria (according to the values of $w_i \mid i = 1..4$) a straightforward sub-optimal solution PELC*(μ^*) can be obtained while using numerical optimization (iterative dichotomy for instance). Regarding the global optimization based on gPELC*, the complexity is slightly increased w.r.t. PELC* planning, but remains controllable (cf. Section 3.2.2).

Among the main interests of the proposed local and global path planning is the possibility to take into account the vehicle kinematic constraints (e.g., its non-holonomy), its initial configuration as well as the used control law. Thus, the performed multi-criteria optimization permits us to lead toward very efficient safe path while permitting to take into account the most pertinent path features: smoothness, length and the precision to reach the final target (given by $J_{\text{Distance\_gPELC\_FinalTarget}}$ in Eq. (12)).

In what follows, a Homogeneous and Hybrid (reactive/cognitive) Control Architecture (HHCA) will be proposed and detailed. This architecture uses notably the definition of optimal PELC* and gPELC* to have a homogeneous way to obtain the vehicle's set-points.

4. Homogeneous and Hybrid Control Architecture (HHCA)

The proposed Homogeneous (in terms of used set-points and control law) and Hybrid (reactive/cognitive) Control Architecture (HHCA, cf. Fig. 6) aims to simplify, manage and control either reactive or cognitive vehicle navigation (cf. Section 1). By reactive (cf. Section 4.4.1), it is mainly meant that the navigation of the vehicle is done with the minimum information on the environment, whereas cognitive control is based on quasi-full knowledge on the environment. Obviously, when this knowledge is available, the cognitive control permits to lead generally to optimal (or sub-optimal) robot navigation. Nevertheless, it needs a lot of processing

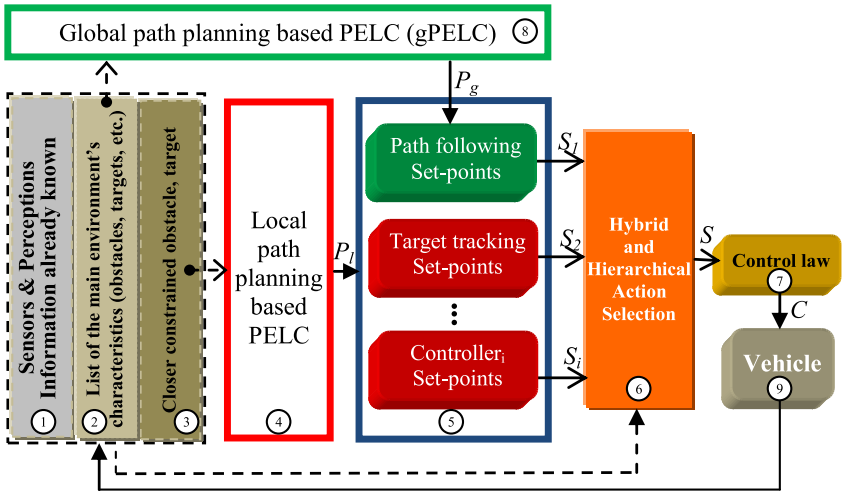


Fig. 6. The proposed Homogeneous and Hybrid Control Architecture (HHCA) for mobile robot navigation.

time to reach this solution which could lead, in certain cases, to an unusable approach (cf. Section 1).

To guarantee the multi-objective criteria linked to the navigation of a vehicle in cluttered environment (cf. Section 1), control architectures can be elaborated in a modular and bottom-up way as introduced in [26] and so-called behavioral architectures [34]. These techniques are based on the concept that a robot can achieve a complex global task while using only the coordination of several elementary behaviors. In fact, to tackle this complexity, behavioral control architecture decomposes the global control into a set of elementary behaviors/controllers (e.g., attraction to a target, obstacle avoidance, trajectory following, etc.) to master better the overall robot behavior.

The proposed HHCA allows, in addition to manage the interactions between different elementary controllers, it permits also to guarantee the stability of the overall control architecture and the smoothness of the vehicle trajectory. The different blocks composing this architecture are detailed below.

4.1. Perceptive and environment characteristics

While using exteroceptive vehicles' sensors and any already known data on the environment, the blocks numbered 1 to 3 in Fig. 6 are in charge of detecting/localizing/characterizing any important features in the environment. Mainly these blocks must provide the list of all perceived obstacles (or known according for instance to a road map) and the target to reach. Any possible obstructing object (obstacle/wall/pedestrian/ etc.) is characterized as specified in Section 2 by an Ellipse given by the parameters (h, k, Ω, A, B) which could be obtained either offline or online.

4.2. Uniform used control law

Before presenting the used control law, it is important to know the robot's model. The used robot corresponds to a tricycle vehicle [69] modeled according to the well-known kinematics model given by Eq. (13).

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = v\tan(\gamma)/l_b \end{cases} \tag{13}$$

where (x, y, θ) is the posture (configuration state) of the vehicle at the point O_m (origin of the local reference frame X_mY_m linked to the vehicle (cf. Fig. 7)), γ is the orientation of the equivalent front wheel (cf. Fig. 7), v is the linear velocity of the vehicle at

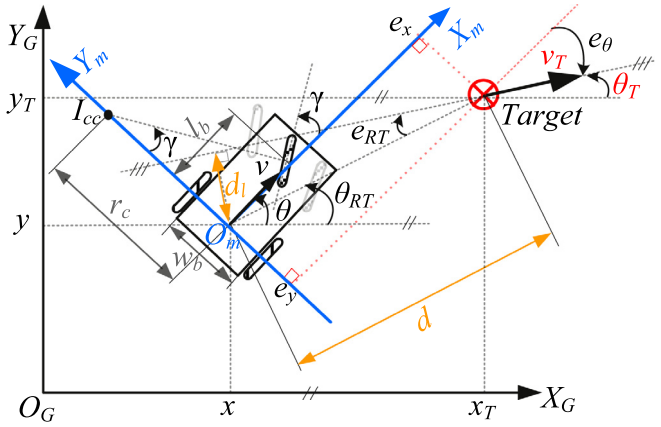


Fig. 7. Vehicle and target configuration in Global and Local reference frames. Control variables according to Lyapunov synthesis are also shown.

O_m and l_b is the vehicle's wheelbase. v and γ are the two control inputs of the vehicle (cf. Eqs. (17) and (18) respectively). According to Fig. 7, w_b corresponds to the track width of the vehicle and I_{cc} the instantaneous center of curvature of the vehicle trajectory. The radius of curvature r_c is given by:

$$r_c = l_b/\tan(\gamma) \tag{14}$$

and $c_c = 1/r_c$ is the curvature of the vehicle trajectory.

The used control law [70] aims to drive the vehicle toward specific targets (static or dynamic) in the environment. At each sample time the tracked target is defined by a posture (x_T, y_T, θ_T) and a velocity v_T (this velocity could be $= 0$ if the target is static). It will be shown in sub-Section 4.3 that different vehicles behaviors will be described with a uniform way where the vehicle has to reach/follow/track a specific target set-points. In order that the paper becomes at maximum self-content, let us give in summary the main elements of synthesis, using a Lyapunov formulation [71], the used control law [70]. The adopted Lyapunov function V is given by Eq. (15) (cf. Fig. 7):

$$\begin{aligned} V &= \frac{1}{2}K_d d^2 + \frac{1}{2}K_l d_l^2 + K_o[1 - \cos(e_\theta)] \\ &= \frac{1}{2}K_d d^2 + \frac{1}{2}K_l d^2 \sin^2(e_{RT}) + K_o[1 - \cos(e_\theta)] \end{aligned} \tag{15}$$

where the initial values of e_{RT} and e_θ must satisfy the following initial conditions [70]:

$$e_{RT} \in]-\pi/2, \pi/2[\quad and \quad e_\theta \in]-\pi/2, \pi/2[\tag{16}$$

The Lyapunov function (15) is therefore a function of three parameters which depend on: the distance d between the target and vehicle's position; the distance d_l from the vehicle to the target line (line that passes through the target position with an orientation equal to the target orientation), this term is related to the Line of Sight and Flight of the target [72]; and the orientation error e_θ between the vehicle and the target.

The desired linear velocity v and the front wheel orientation γ of the vehicle which permits to asymptotically stabilize the error vector $(e_x, e_y, e_\theta, (v - v_T))$ toward zero (permitting therefore to have $\dot{V} < 0$) are given by:

$$v = v_T \cos(e_\theta) + v_b \tag{17}$$

$$\gamma = \arctan(l_b c_c) \tag{18}$$

where v_b and c_c are defined by:

$$v_b = K_x [K_d e_x + K_l d \sin(e_{RT}) \sin(e_\theta) + K_o \sin(e_\theta) c_c] \tag{19}$$

with:

$$c_c = \frac{1}{r_{c_T} \cos(e_\theta)} + \frac{d^2 K_l \sin(e_{RT}) \cos(e_{RT})}{r_{c_T} K_o \sin(e_\theta) \cos(e_\theta)} + K_\theta \tan(e_\theta) + \frac{K_d e_y - K_l d \sin(e_{RT}) \cos(e_\theta)}{K_o \cos(e_\theta)} + \frac{K_{RT} \sin^2(e_{RT})}{\sin(e_\theta) \cos(e_\theta)} \tag{20}$$

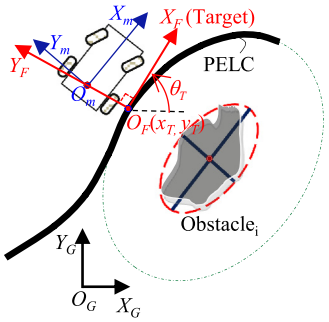
$\mathbf{K} = (K_d, K_l, K_o, K_x, K_\theta, K_{RT})$ is a vector of positive constants defined by the designer. Accurate analysis of this stable and efficient control law is given in [70] and [5].

Knowing the used common control law permitting to reach any static or dynamic target with stable way, let us present in what follows, how to perform different navigation sub-tasks using appropriate and common definition of target set-point configurations $(x_T, y_T, \theta_T, v_T)$.

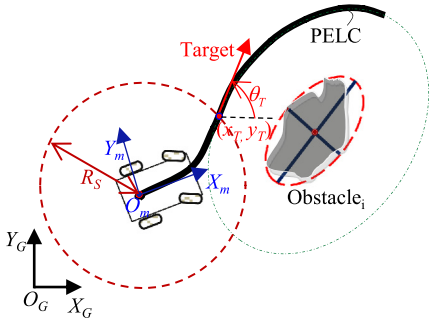
4.3. Sub-tasks to achieve based on homogeneous set-points definition

As described in the introduction of Section 4, the design of bottom-up approach requires to decompose the overall complex task into a multitude of sub-tasks to achieve (e.g., obstacle avoidance, wall following, target reaching, etc.). According to the elementary sub-task to achieve in reactive or cognitive way (cf. Section 4.4) it has been noted in general that the vehicle has to either follow/track a path/trajectory or reach/track a static/dynamic target to perform the assigned sub-task. As examples of tasks definition using target reaching or path following, one can cite: safe vehicle navigation in urban environment through pre-defined static targets [5] or the task consisting to maintain a formation with a group of robots while using dynamic virtual targets [73]. Still for multi-robot formation task, the Leader while avoiding safely an obstacle, transmits its trajectory which must be tracked by the Followers [74].

The two main behaviors (path following and target reaching) will be shown throughout respectively Sections 4.3.1 and 4.3.2. These two subsections will give a homogeneous way to define different set-points $(x_T, y_T, \theta_T, v_T)$. It is noticed that the set-points configurations are taken within PELC-generated trajectories. Indeed, the proposed HHCA architecture contains different set-point blocks which have as input the PELC already defined either locally (block number 4 in Fig. 6 for reactive navigation (cf. Section 4.4.1)) or globally (block number 8 in Fig. 6 for cognitive navigation (cf. Section 4.4.2)). Once these set-points are obtained, the common control law defined in Section 4.2 is used to stabilize the error to zero.



(a) Target based on Frenet frame.



(b) Target based on Euclidean distance.

Fig. 8. Target set-points definition w.r.t. (a) Frenet reference frame or (b) Euclidean distance.

4.3.1. Path following set-points based on Frenet reference frame

Once a path is obtained using PELC, in certain situations (e.g., static environment) it is enough for the vehicle to follow it as accurate as possible without modifying this initial planning. In that situation, it is used a Frenet reference frame [10] to extract the vehicle's set-points. The target set-point, at each sample time, is given by (cf. Fig. 8(a)):

- A position (x_T, y_T) corresponding to the closest position in the pre-planned PELC w.r.t. the origin of the reference frame $X_m Y_m$. (x_T, y_T) point corresponds to the origin of Frenet reference frame $X_F Y_F$.
- An orientation θ_T corresponding to the tangent of the PELC w.r.t. $X_G Y_G$ reference frame.
- A velocity v_T which could be constant or variable indifferently.

4.3.2. Target reaching/tracking set-points

When the environment is not very well known or dynamic, it is better to navigate reactively (cf. Section 4.4.1). In that situation, the current PELC takes as initial configuration, at each sample time, the current vehicle configuration. The target set-point is given, at each sample time, by (cf. Fig. 8(b)):

- A position (x_T, y_T) corresponding to the intersection between the circle (which has as origin the reference frame $X_m Y_m$ and as radius R_S) and the planned PELC.
- An orientation θ_T corresponding to the tangent to the PELC w.r.t. $X_G Y_G$ reference frame at the intersection point (x_T, y_T). If $R_S = 0$, the vehicle has to apply only an orientation control. Indeed, since the robot is already on the current computed limit-cycle (e_x and e_y will be equal therefore to 0 in Eqs. (17) and (18)), the robot has to only control its heading w.r.t. θ_T. This simple case of reactive control is shown in [55,57].
- A velocity v_T which could be constant or variable indifferently.

4.4. Reactive versus cognitive vehicle navigation

It is shown in what follows the different proposed navigation strategies based on PELC. The focus will be given mainly for:

- the way to act online to changed local environment when the used strategy is reactive (cf. Section 4.4.1),
- the used cognitive navigation (cf. Section 4.4.2).

In Section 4.4.3 the choice between either reactive or cognitive modes will be illustrated through the proposed hybrid and hierarchical action selection process.

4.4.1. Reactive navigation based on local PELC*

Reactive navigation is most appropriate when the environment is not well known or in the case where this environment is highly dynamic and/or uncertain. The vehicle aims to navigate from its initial position to the final target while using only local defined PELC* (cf. Section 3.2.1), dealing therefore sequentially with obstructing obstacles/walls/etc. This reactive navigation supposes nevertheless the efficiency and the reliability to obtain online the features of the obstacles (cf. Section 2). The vehicle has to react therefore online to its environment according only to its current available knowledge. It is why this kind of navigation could not ensure the optimality of the global navigation (cf. Section 4.4.2).

The applied reactive navigation, detailed in Algorithm 2, activates the obstacle avoidance behavior as soon as there exists at least one object which can obstruct the future robot movement toward its target. Otherwise, the behavior of target reaching (still using PELC*, cf. Section 3.2.1) is activated. It is mentioned in Algorithm 2, the notion of "Current final target" because the vehicle is supposed to have a multitude of sequential static targets to reach [5]. The good performance of the reactive navigation needs to manage some conflicting situations which could, in certain cases, lead to trajectory oscillations or dead-ends. Several reactive rules are detailed in [55] to avoid these situations. For instance, it has been proposed to maintain the direction of avoidance (clockwise or counter-clockwise) when the robot avoids two consecutive obstacles (without finishing yet the avoidance of the first, therefore the vehicle does not yet reach the first obstacle axis Y_{OT} (cf. Fig. 4)).

Algorithm 2: Reactive navigation using optimal PELC* paths

Input: All the features h, k, Ω, A, B of the closest constrained obstacle (cf. Algorithm 3); Value of K_p (the desired minimum safe distance "offset" to the obstacles); Current final target localization (x_f, y_f).
Output: Current PELC*$((h, k), \Omega, (A, B, K_p), \mu^*)$ to follow (cf. Equation (3) and Section 3.2.1).

if *It exists at least one obstructing obstacle (cf. Algorithm 3)* **then**
  //Obstacle avoidance behavior
  Obtain μ^* optimizing PELC according to the criteria J (cf.section 3.2.1).
  PELC*$((h, k), \Omega, (A, B, K_p), \mu^*)$;
else
  //Target reaching behavior
  PELC*$((x_f, y_f), 0, (\xi, \xi, \xi), \mu^*)$;
  Where ξ is a positive very small value $\longrightarrow 0$
end

Algorithm 3: Obtaining the most obstructing obstacle

Input: All the features $(h, k, \Omega, A, B)_i$ (cf. equation (1)) of the detected Obstacles in the vehicle's field of view; Value of K_p (the desired minimum safe distance "offset" to the obstacles); Current final target localization (x_f, y_f).
Output: The index "k"of the most obstructing obstacle (if it exists).

for *Each Obstacle$_i$* **do**
  if *The obstacle is an obstructing obstacle {i.e., it exists one intersect point between the line "l" (connecting the vehicle to the target (cf. Figure 1)) and the Parallel Ellipse of Influence of the Obstacle$_i$* **then**
    Add the *Obstacle$_i$* to *ListObstructingObstacles*;
  end
end
if *ListObstructingObstacles* $\neq \emptyset$ **then**
  Extract the index $k \in$ *ListObstructingObstacles* of the closest obstacle (in terms of Euclidean distance to the vehicle (D_{ROi} in Figure 1)).
else
  There is not obstructing obstacle, the environment is safe.
end

4.4.2. Cognitive navigation based on global PELC* (gPELC*)

To perform cognitive navigation, it implies obviously much more knowledge about the environment (generally all the free and obstructing spaces, called respectively C_{Free} and $C_{Obstacles}$). In this kind of navigation, the robot must define an overall path/trajectory while taking into account the possible multiple obstacles in the environment. Indeed, the navigation could be optimal to avoid a single obstacle using an appropriate PELC* (cf. Section 3.2.1) but not optimal at all if the navigation strategy needs to take into account several obstacles before reaching the final target [75]. To perform cognitive navigation, the proposed HHCA uses a global optimized paths obtained according gPELC* (cf. Section 3.2.2). Several simulations will be shown in Section 5.1.

4.4.3. Hybrid and hierarchical action selection

In multi-controller architectures as what is proposed in this paper, there are exists two major principles for controller/behavior coordination: *action selection* and *fusion of actions* which lead respectively to competitive and cooperative control architectures. In competitive architectures (*action selection*), the set-points sent to the robot's actuators at each sample time are given by a unique controller (behavior) which is selected among a set of possible controllers. The principle of competition can be defined by a set of fixed priorities like in subsumption architecture [26] where a hierarchy is defined among the behaviors. The *action selection* can also be dynamic without any hierarchy between behaviors [76,77]. In cooperative architectures (*fusion of actions*), the set-points sent to the robot's actuators are the result of a compromise or a fusion between controls generated by several active behaviors. These mechanisms include fuzzy control [78] *via* the process of defuzzification, or multi-objective techniques to merge the controls [79]. Among these cooperative architectures, schema-based principle [27] is among the ones which has an important impact in the scientific community. Even if the *fusions of actions* process gives a very interesting robot's behaviors, the stability of the overall control architecture is generally very hard, even impossible to demonstrate. Contrary, control architectures based on *action selection* process are usually much more easy to demonstrate their overall stability even when switch between the behaviors occur [31,49].

Algorithm 4: Hybrid and Hierarchical Action Selection process

```
Data: Environment knowledge and perceptive information
Result: The more appropriate navigation strategy
while Final target is not reached do
    if gEPLC* exits then
        //Cognitive navigation
        //gEPLC* exits means that the overall environment
        //knowledge is available
        Path following control activation w.r.t. gPELC*;
    else
        //Reactive navigation
        //Defined w.r.t. the current obtained PELC*
        if Static obstacles & certain environment then
            Local path following control activation;
        else
            Target reaching control activation;
        end
    end
end
```

The proposed multi-controller architecture (cf. Fig. 6) is based on the *action selection* process. It is called the Hybrid (reactive/cognitive) and Hierarchical Action Selection process and is summarized in Algorithm 4. This process aims to activate either reactive or cognitive navigation (cf. sub-Sections 4.4.1 and 4.4.2) according to the environment knowledge and perceptions.

The cognitive navigation is activated only if the entire environment is well known or when the navigation is achieved in relatively low dynamic environment; low enough so that the gPELC (cf. Section 4.4.2) could be re-computed online. In the case where the gPELC is impossible to obtain online, instead of stopping the vehicle's navigation (which could be still an option), the vehicle will switch to navigate in a reactive way. This last navigation could be done in two ways: the first consists of using path following control, based on local computed PELC, in the case where the current obstructing obstacle is static and could be accurately detected (cf. Section 4.1); the second reactive navigation is performed if the environment is highly dynamic and/or with a lot of uncertainty, in this case the vehicle has to navigate with even more reactivity (no pre-planned path to follow), using online target reaching control (cf. Section 4.3.2). The R_S radius given in Section 4.3.2 could be fixed according to the measured uncertainty rate and to the dynamicity of the obstacles. This implies obviously to have specific metrics to characterize the environment uncertainty and the dynamicity of the detected obstacles (velocity, acceleration, etc.). These two criterion are more linked to the perceptive aspect of the control architecture and will not be addressed in this paper.

5. Proposal validation

5.1. Extensive validation by simulation

Once presented the most important blocks characterizing the proposed control architecture, let us show in what follows several simulations' examples to exhibit the architecture's flexibility and efficiency (cf. Section 1). The simulations were implemented using MATLAB® software (the porting to C++ language will be done in the near future to enhance the processing time) and performed with an Intel Core I7, CPU of 2.70 GHZ and a RAM of 32 GO. As given in Section 4.2, the robot kinematics is based on a tricycle model (cf. Fig. 7); its features and control parameters are given by:

- l_b = 12 cm and γ_{max} = 45°.
- To characterize the robot collisions with the environment, the robot is surrounded completely by an ellipse (which has a major axis of 14 cm and a minor axis of 9 cm).
- The robot maximum field of view is considered as a circle centered on the robot with a radius of 72 cm (cf. Fig. 13). This circle corresponds to the maximum distance where the robot can detect an obstacle. This perception is mainly used for safety behavior (automatic stop if the obstacle is too close) or for reactive navigation.
- The control law parameters are given by $\mathbf{K} = (10, 5, 2, 0.3, 5, 0.01)$ (cf. Section 4.2).

The first set of simulations (cf. Section 5.1.1) will show the use of gPELC* for different kind of environments (cluttered, structured, etc.) and the other set of simulations (cf. Section 5.1.2) will show the use of PELC and gPELC* to perform either reactive or cognitive navigation according to the environment state/context.

5.1.1. The use of gPELC* for different environments

Cluttered environment. Fig. 9(a) shows the application of Algorithm 1 for a set of μ values $S_\mu = \{0.1, 0.4, 0.7\}$ in each direction (clockwise and counter-clockwise). Several simulations with their initial inputs and results are summarized in Table 1. This table shows, for instance, the weights $W_{gPELC} = \{w_1, w_2, w_3, w_4, w_5\}$ characterizing the global cost function (cf. Eq. (12)) for each planned gPELC, the obtained Tree characterizing the optimization process. In the simulation given in Fig. 9(a), Algorithm 1 had to explore 31 vertices which required 9.48 s as computation time. This optimization permits to obtain 14 valid gPELC and the optimal one (according to the applied global cost function) has an optimal cost of $G = 3.03$.

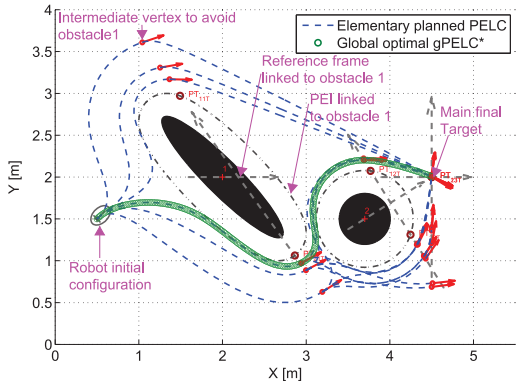
Fig. 9(b) corresponds to a simulation which has the same initial inputs as in Fig. 9(a) but while Expanding the tree (given by Algorithm 1) for only the optimal vertices (cf. Table 1). According to this table, it is found that the computation time is reduced by 28%, and despite this, the obtained optimal gPELC* cost G is still very close to the obtained gPELC* given in Fig. 9(a). The obtained gPELC* is nevertheless different in terms of path shape and intermediate cost-function (as for G_2, where the obtained gPELC* is smoother than the one obtained in Fig. 9(a) ($G_2 = 0.18$ instead of 0.25). Fig. 10(a) shows the obtained Tree related to the simulation given in Fig. 9(b). Due to the smaller number of obtained vertices while using only optimal vertices expansion, this methodology will be preferred in the coming simulations to highlight better certain results. Fig. 9(c) gives an example of the influence of the wights chosen for W_{gPELC} to obtain the optimal gPELC*. Indeed, while modifying these weights w.r.t. the simulation given in Fig. 9(b), the obtained final gPELC* has been changed (cf. Table 1).

While simulations given in Fig. 9(a) to 9(c) show the application of Algorithm 1 for a quite simple environment (2 obstacles), the simulations given in Fig. 10(b,c) permit, among other things, to highlight the efficiency of the proposed Algorithm 1 for an even more cluttered environment (a configurations of 5 close obstacles) (cf. Table 1). These last simulations will be used in next Section 5.1.2, where the obtained optimal path gPELC* will be used as an initial path to show the flexibility of the overall proposed control architecture to switch easily and safely from cognitive navigation to a reactive one and *vice versa*.

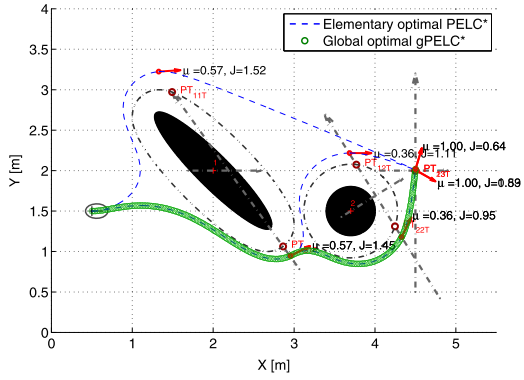
Structured environment. To highlight the reliability of the proposed planning method-based on PELC for different kinds of environments, it is also important, in addition to a cluttered environment, to verify its efficiency for a structured environment (as we can find in an indoor or urban environment with walls and right angles "perpendicular straight lines"). The obstacle modeling given in Section 2 is used in what follows to surround walls (or sidewalks for instance) of different dimensions (cf. Fig. 11). Each obstacle is

Table 1
Algorithm 1 parameters and optimization results to obtain optimal gPELC* for different environments.

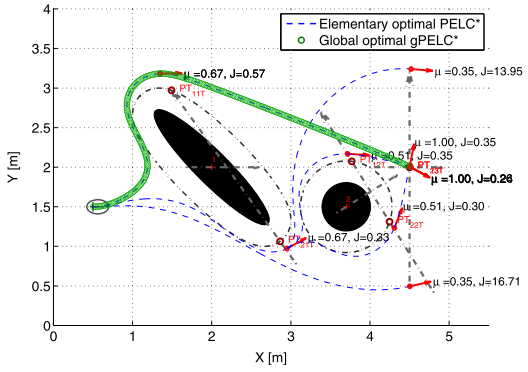
Figure	Exhaustive expanding tree?	W_{gPELC}	Nbre of valid explored vertices	Nbre of valid gPELC	Time [s]	Optimal gPELC* cost function			
						G	G_1	G_{1Bis}	G_2
9(a)	Yes	{ 0.50, 0.0005, 10^6, 10^6, 10}	31	14	9.48	3.03	0.21	2.57	0.25
9(b)	No	{ 0.50, 0.0005, 10^6, 10^6, 10}	7	3	6.85	3.04	0.35	2.51	0.18
9(c)	No	{ 0.95, 0.0005, 10^6, 10^6, 10}	10	5	7.72	0.83	0.32	0.29	0.22
11(a)	No	{ 0.60, 0.0020, 10^6, 10^6, 10}	5	2	15.68	3.69	0.26	2.69	0.74
11(b)	No	{ 0.50, 0.020, 10^6, 10^6, 10}	9	2	6.19	11.92	1.60	6.55	3.77
11(c)	No	{ 0.50, 0.020, 10^6, 10^6, 10}	14	4	6.36	11.40	1.02	6.65	3.73
10(b)	Yes	{ 0.60, 0.0020, 10^6, 10^6, 10}	49	18	82.12	4.91	0.58	3.45	0.88
10(c)	No	{ 0.60, 0.0020, 10^6, 10^6, 10}	21	5	49.63	5.00	0.64	3.42	0.94
12(a)	No	{ 0.60, 0.0020, 10^6, 10^6, 10}	13	5	28.07	3.46	0.42	2.33	0.71



(a) Exhaustive Expanding Tree (cf. Algorithm 1). This sub-figure highlights also the main shape features of the following simulations.



(b) Expanding Tree only for optimal node at each step.



(c) Expanding Tree only for optimal node at each sample time with other criteria parameters w.r.t. simulation (b).

Fig. 9. Global path planning based on gPELC*. Gray "−.−" lines correspond to the PEI_i linked to each $obstacle_i$.

surrounded with thin Surrounded Ellipse (given by SE_i in Fig. 1) and with a Parallel Ellipse of Influence (PEI_i in Fig. 1) which will allow a safe margin between the robot and the considered obstacle. The first simulation given in Fig. 11(a) shows the efficiency of Algorithm 1 even for a Trap configuration. The robot initial posture is $(x_0, y_0, \theta_0, \gamma_0) = (0, 1.5, -45°, 0°)$. Algorithm 1 obtained an optimal gPELC* (cf. Table 1) while avoiding locals minima [35]. According to Table 1, it is seen that the time necessary to obtain gPELC* is equal to 15.68 s which is relatively high w.r.t. other simulations which have only few valid explored vertices. This is explained by the fact that according to the configuration given by this trap, a lot of iterations of Algorithm 1 leads to invalid vertices (thus, collision of the computed PELC with an obstacle).

The simulations given in Fig. 11(b) and 11(c) show a complex enough environment based on several walls/obstacles (forming labyrinthine corridors). The setup difference between the simulation given in Fig. 11(b) and the one given in Fig. 11(c) is in the value of the obstacle safe Margin (cf. Section 3.1.1) which corresponds to K_p (cf. Fig. 1 and Eq. (3)). For the first simulation (cf. Fig. 11(b)) $K_p = 16$ cm whereas the second (cf. Fig. 11(c)) $K_p = 32$ cm (cf. Table 1 for simulations parameters and optimization results). The second simulation restricts much more the possible robot states to reach the final Target, but permits to obtain much more safe gPELC* because the robot is forced to navigate as fare as possible from the obstacles. The obtained gPELC* is close to one path which could be obtained by a Voronoï method [12,13], but with an important advantage in what we propose is that the obtained path (gPECL*) takes into account:

- the kinematic and structural constraints of the robot (non-holonomy, maximum steering angle γ_{Max}, etc.),
- multi-criteria optimization (cf. Eq. (12)) and not only a safety criterion as for Voronoï method.

It is to be noted that K_p attributed to each obstacle could be different between them. This could be chosen according, for instance, to the obstacle's dynamic or to its features (location, shape, etc.). It is to be noted also that in certain situations where the PEI_i attributed to $obstacle_i$ is in intersection with the PEI_j of another $obstacle_j$ (as seen in Fig. 11), it is important to introduce some simple rules to guarantee always the convergence of Algorithm 1. Indeed, when computing an elementary PELC w.r.t. an $obstacle_i$, the axis Y of the reference frame linked to this obstacle (cf. Section 3.1.2) could not be reachable without going inside PEI_j. This is due to the intersection of the PEI (attributed to the two or more obstacles). The condition of stopping the avoidance w.r.t. an obstacle can never therefore be attained. Thus, if at least two obstacles have an intersected PEI, the following simple rules, to end the computation of the $PELC_i$, must be applied:

- End (stop and validate) the computed $PELC_i$, the first time that its x abscissa sign (w.r.t. the reference frame $\Re_i$ linked (∞) to $obstacle_i$) changes from "−" to "+".

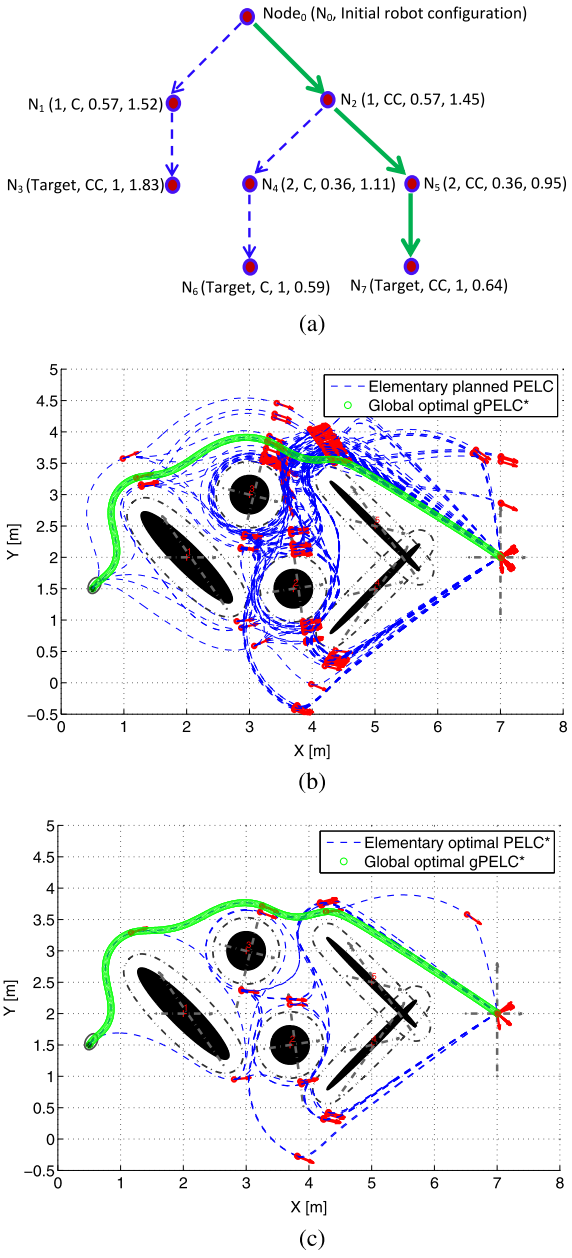


Fig. 10. (a) Tree representation to obtain the gPELC* given in Fig. 9(b). Each node $N_i|_{i=1..7}$ (except the root node N_0) is represented respectively by: an index 1 or 2 (the $_{id}$ of the avoided obstacle) or Target (for final target reaching); the direction of avoidance C or CC (for respectively Clockwise or Counter-Clockwise); the value of μ^* and finally the value of the elementary obtained PELC* cost J^* (cf. Eq. (8)). The green arrows correspond to the optimal solution. (b) and (c) Global path planning based on gPELC* using, respectively, Exhaustive Expanded Tree and only optimal ones (at each step). Gray "−.−" lines correspond to the PEI_i linked to each $obstacle_i$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

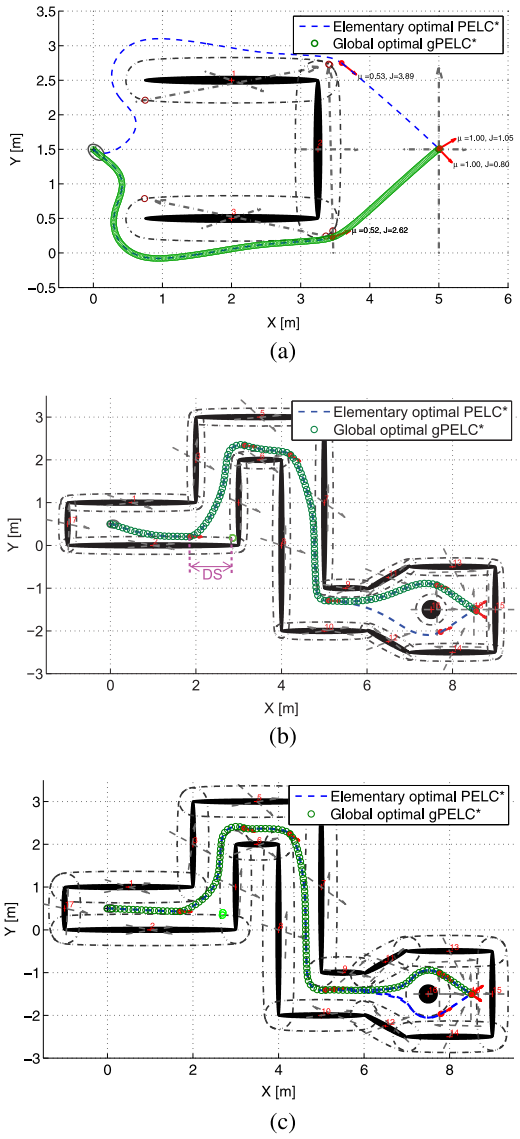
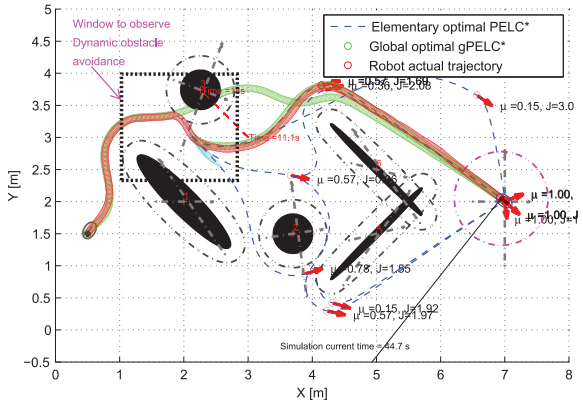


Fig. 11. (a) Efficient gPELC* even for a trap configuration. (b) and (c) Global path planning-based gPELC* in structured environment, with respectively: (b) K_p = 16 cm and (c) K_p = 32 cm. Gray "−.−" lines correspond to the PEI_i linked to each $obstacle_i$.

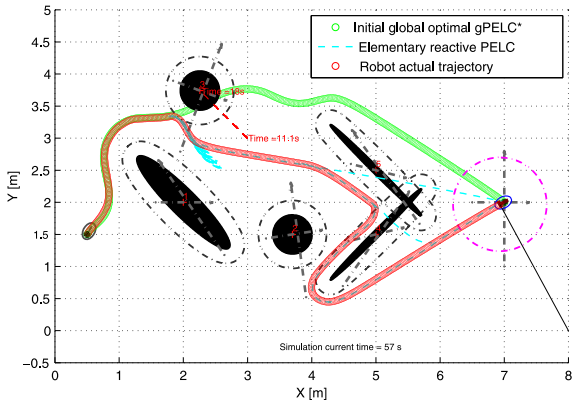
- If the above rule is not yet verified and the current computed $PELC_i$ is going to be inside another PEI_j (with $j \neq i$), therefore, stop the computation of $PELC_i$ at a predetermined distance DS before going inside PEI_j, a node is therefore added in Algorithm 1, permitting to continue the expanding while avoiding a deadlock. For example, in Fig. 11(b), the robot must avoid the $obstacle_2$ in a clockwise direction, but its PEI_2 intersects with PEI_4, the above rules were therefore applied. It is to be noted that this conflicting situation, where two PEI intersects, has been resolved in [55] for reactive navigation while the avoidance switching from one obstacle to another is activated as soon as the second obstacle becomes the closest to the robot.

5.1.2. Switch from cognitive to reactive navigation and vice versa

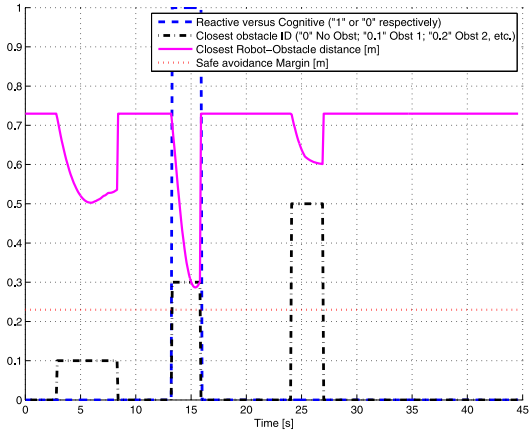
The aim of the following simulations is to show the flexibility of the proposed control architecture to perform either reactive or cognitive navigation (cf. Section 4.4) while guaranteeing smooth and steady robot behavior (cf. Section 4.2). Thus, using a relatively simple algorithm (as given in Algorithm 4), the possibility of switching from cognitive to reactive mode and *vice versa* will be shown. It is to be noted that these simulations do not aim to propose an optimal choice between the activation of one mode (cognitive or reactive) w.r.t. the other. This important issue is an open research area [38–40,18] and the inherent structure of the proposed control architecture is particularly appropriate to address this kind of interesting issue. This will be the subject to future developments.



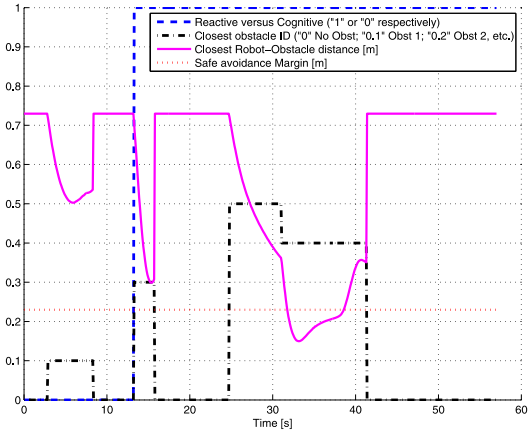
(a) Cognitive → Dynamic obstacle avoidance → Cognitive navigation.



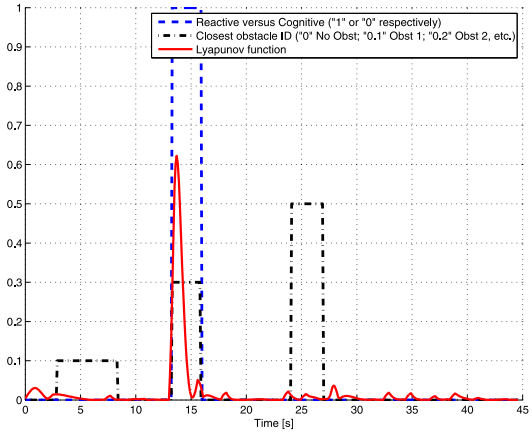
(b) Cognitive → Dynamic obstacle avoidance → Reactive navigation.



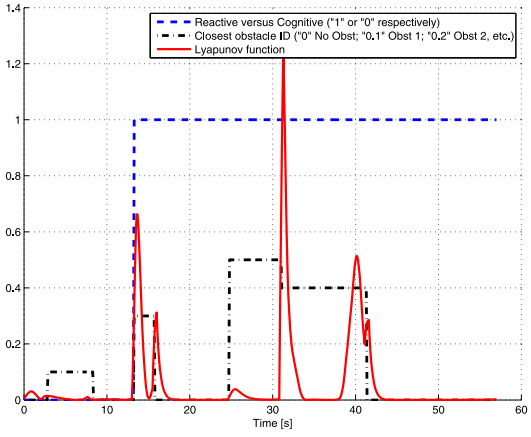
(c) Simulation indicators.



(d) Simulation indicators.



(e) Lyapunov function progress.



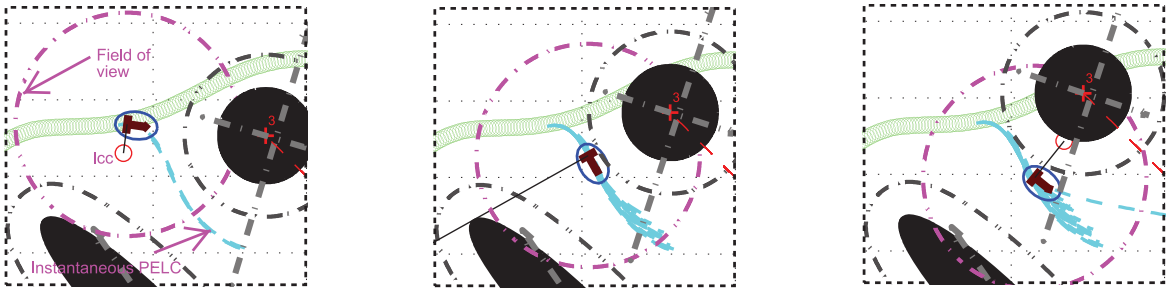
(f) Lyapunov function progress.

Fig. 12. Reactive versus cognitive navigation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The robot navigation given in Figs. 12(a) and 12(b) show the activation of different modes. The sequence given in Fig. 12(a) is Cognitive → Dynamic obstacle avoidance and finally Cognitive navigation, whereas in Fig. 12(b): Cognitive → Dynamic obstacle avoidance → Reactive navigation. In both simulations, the robot starts by performing path following (cf. Section 4.3.1) of the already obtained gPELC* given in Fig. 10(b) (cf. Table 1). Since the initial obtained gPELC* considers only static environment, the robot starts to follow this global optimal path to reach the final Target. At the instant 11 s, the obstacle$_3$ starts to move in a straight line (cf. Fig. 13(a) to 13(c)). This movement yields to make unsure the initial gPELC*, therefore, as soon as this obstacle is in the field of view of the robot (represented by pink dashed circle in Fig. 13), the robot starts to perform reactive obstacle avoidance using local PELC (cf. Section 4.4.1). The used set-points in this navigation phase are based on target reaching set-points (cf. Section 4.3.2) while taking the parameter $R_S = 0$ (cf. Fig. 8(b)).

Once obstacle$_3$ is completely avoided (cf. Algorithm 2), the robot continues to reach the final target by using either cognitive navigation (cf. Fig. 12(a))[2] or reactive navigation (cf. Fig. 12(b)). In the first case, a new gPELC* is recomputed (from the current initial

[2] cf. "https://goo.gl/iZ8YQA" to show the video of the corresponding simulation.



(a) Time = 13.5 s. (b) Time = 15 s. (c) Time = 16 s.

Fig. 13. Dynamic obstacle avoidance. The cyan dashed lines in (a) to (c) correspond to the different local re-computed PELC (at each sample time "updated robot's location/perception") by HHCA when the robot avoids Obstacle$_3$. In (c) Obstacle$_3$ is completely avoided and a PELC is computed w.r.t. the main target.

robot's configuration) and followed by the robot. The re-planed path features are given in the last row of Table 1. Figs. 12(c) and 12(e) give the robot's navigation details of the simulation given in Fig. 12(a), whereas Figs. 12(d) and 12(f) give the navigation's details of Fig. 12(b).

Concerning the simulations' indicators given in Figs. 12(c) and 12(d), they show in general that in both simulations, the robot navigates far enough from the closest obstacle. Indeed, specifically in Fig. 12(c), the distance progress between the robot and the closest obstacle is always smooth and bigger than the fixed safe avoidance offset ($K_p = 22$ cm and represented in dotted red lines in the figures). There exists nevertheless one critical phase in Fig. 12(d) where the robot, in reactive mode, is currently finishing to avoid obstacle$_5$ and starting to avoid obstacle$_4$. In this phase the distance robot-obstacle$_4$ is less than the offset but the robot remains far enough from the obstacle to avoid any collision (cf. Fig. 12(b)). In fact, the offset value is fixed according: first, to the dimension of the robot (surrounded by an ellipse with a major axis equal to 14 cm) and also to the robot's physical constraints and control reliability, etc. In the simulation given in Fig. 12(d), the value of the minimum distance robot-obstacle$_4$ is equal to 15 cm, and therefore, in all cases, bigger than the major axis of the surrounded ellipse (i.e., robot-obstacle do not collide).

It is to be noted also that this critical situation happened because of the difficult initial robot configuration when obstacle$_4$ must be avoided. Indeed, when obstacle$_4$ becomes the most obstructing obstacle (cf. Algorithm 3), the robot is very close to it and has an important angular error to the PELC set-point $\geq 90°$, therefore the robot is in a very difficult configuration to safely avoid obstacle$_4$. These specific critical situations are unfortunately unavoidable in reactive mode since the robot discovers its environment online [55,57]. An emergency stop could be obviously activated if the robot-obstacle distance is less than a certain value. In addition, as information, the computed PELC (to avoid obstacle$_4$) has a counter-clockwise direction to avoid obstacle$_4$. Thus, in the same direction then the avoidance of obstacle$_5$. Indeed, in reactive avoidance, if the robot starts to avoid an obstacle$_i$ and switches to another obstacle$_j$ (which has an intersection w.r.t. obstacle$_i$), the robot must follow the same direction (clockwise or counter-clockwise) as for the previous avoidance. This avoids the robot's dead-ends and infinite oscillations [55].

Figs. 12(e) and 12(f) give the progress of the Lyapunov function (cf. Eq. (15)). These figures show the stability of the proposed control to make the errors converge always to 0, even when the robot enters reactive mode, where the obstacle to avoid can change suddenly according to the robot's perceptions (cf. Algorithm 3).

In fact, when the robot starts to avoid another obstacle, another local PELC will be re-computed, taking into account the new current obstacle features (position, orientation, dimension, etc.). This causes inevitably an abrupt jump in the error value (therefore on the Lyapunov function), but after that, this function decreases always until reaching 0, which attests to the asymptotic stability of the overall control architecture.

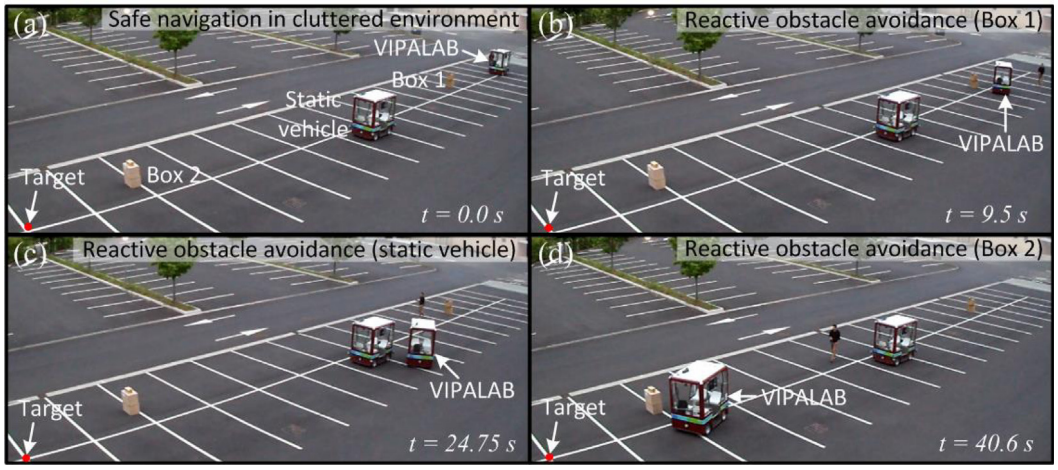
5.2. Experimental validation

This subsection describes the first performed experiment using an urban electric vehicle, called VIPALAB® [80] (cf. Table 2 for some VIPALAB's specifications). It is to be noted that this vehicle is designed to perform autonomous navigations mainly in narrow and highly dynamic urban environments (used for instance as an autonomous shuttle between several locations in a midtown). This vehicle should therefore navigate even in the presence of a lot of pedestrian around, which means, in all cases, while having a relatively slow velocity (at maximum of 5.5 m/s, as indicated in Table 2). In the presented experiment, the VIPALAB has to reach a predefined final static target while avoiding three static obstacles (two boxes and one static VIPALAB vehicle, cf. Fig. 14). For safety reasons it was chosen, in this first experiment, to constrain the velocity of the platform to 1 m/s. A LIDAR mounted in the front of the vehicle is used for online obstacle detection. Each detected obstacle is surrounded by an ellipse as given in Section 2. More precisely each Surrounded Ellipse parameters (h, k, A, B and Ω, cf. Eq. (1) and Fig. 1) were reliably and smoothly obtained using online range data [50–52]. It is to be noted that the vehicle trajectory has been recorded using an RTK-GPS mounted on the VIPALAB vehicle. This RTK-GPS has been also used to localize the vehicle w.r.t. the final target to reach.

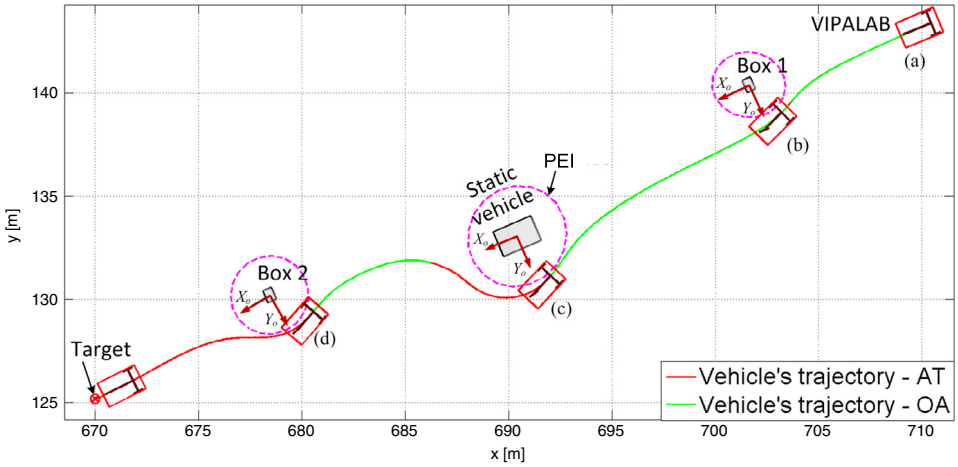
Fig. 14(a) shows some images of the performed experiment.[3] This autonomous navigation has been obtained using the proposed control architecture (cf. Section 4) while using only its fully reactive navigation functionalities. The computed limit-cycles for each avoided obstacle have the same μ values (cf. Eq. (3)). Fig. 14(b) shows the overall actual vehicle's trajectory (and not the generated set-points paths based on elliptic limit-cycles). It can be observed that the vehicle avoids the 3 obstacles with a smooth way and converges safely toward its target. The trajectory of the vehicle between the point (c) and (d) shows an important reorientation amount, this is mainly due to the vehicle's kinematic constraints (e.g., non-holonomy, maximum angular velocity) and to the features of the used non-linear control law (cf. Section 4.2).

Fig. 15(a) shows the evolution of the control law values (v and γ, cf. Section 4.2) and the actual vehicle's outputs. It is observed that the vehicle's velocity decreases according to the distance to the obstacles which allows to enhance the navigation smoothness and safety. Fig. 15(b) presents the progress (during the navigation task) of the Lyapunov function V (cf. Eq. (15)) related to

[3] cf. "https://goo.gl/L96eQK" to show the video of the performed experiment.



(a)



(b) AT and OA correspond respectively to the activation of "Attraction to the Target" or "Obstacle Avoidance" controllers.

Fig. 14. (a) Some images from the performed experiment (b) Vehicle's trajectory using the simplified version of the proposed control architecture (reactive mode).

Table 2
VIPALAB platform.

VIPALAB	Elements	Description
	Chassis	(l, w, h) = (1.96,1.30,2.11)m
	Weight	400 kg (without batteries)
	Motor	Triphase 3x28 V, 4 KW
	Break	Integrated to the motor
	Maximum speed	20 km/h (≈ 5.5 m/s)
	Batteries	8 batteries 12 V, 80 Ah
	Autonomy	3 hours at full charge
	Computer	Intel Core i7, CPU:1.73 GHz RAM: 8Go OS(32 bits): Ubuntu12.04

the used control law. This function decreases asymptotically which attests on the stability of the system.

Therefore, this elementary experiment shows a reliable and efficient reactive navigation in a cluttered environment. Obviously, it does not show all the important features of the overall proposed multi-controller architecture (cf. Fig. 6), which are nevertheless shown extensively in simulation (cf. Section 5.1), but shows at least the viability of this kind of navigation based on limit-cycles. Much more experiments will be performed in future in order to verify the main characteristics of the proposed control architecture (with its different components). Among the most important targeted experiments in near future correspond to perform online cognitive navigation based on gPLEC* and to observe the limits of the proposed control in terms of maximal vehicle dynamic, which must still ensure safe navigation. This will be done as soon as Algorithm 1 will be embedded on the VIPALAB and could be performed in real time (by minimizing mainly the computation time to find the optimal gPELC*). An appropriate control law, which takes into account the dynamical model of the vehicle, could be used to better master the trajectories of the vehicle having much higher velocities and/or accelerations.

6. Conclusion and further work

This paper proposed an overall control strategy (planning–decision–action and their interactions) for autonomous vehicles

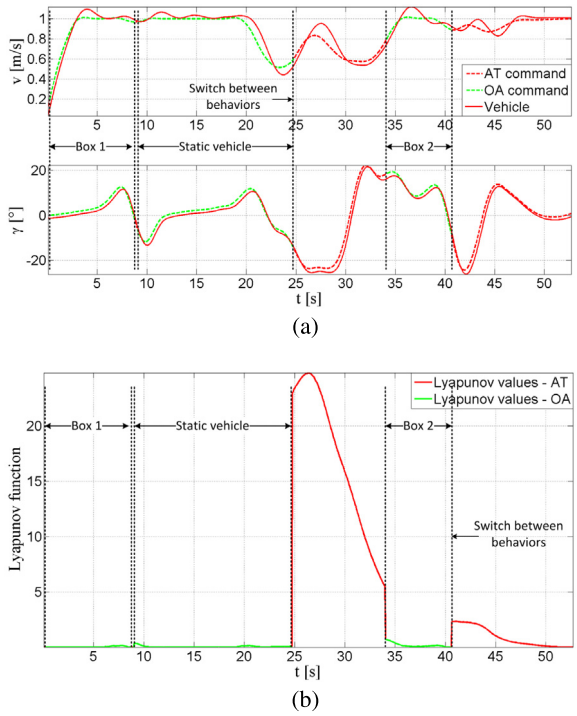


Fig. 15. AT and OA correspond respectively to the activation of "Attraction to the Target" and "Obstacle Avoidance" controllers, (a) Commands given by the control law (b) Evolution of Lyapunov function during the vehicle navigation.

navigation in different environment contexts (e.g., cluttered or not, dynamic or not, etc.). It introduces an original multi-controller architecture (called HHCA, for Homogeneous and Hybrid Control Architecture) and its main components. Among them the ones related to short and long term planning using new formulation of limit-cycles (called PELC, for Parallel Elliptic Limit-Cycle) and their use for hybrid (reactive or cognitive) navigation. More precisely, it is presented first in the paper the mathematical formulation of PELC and its advantages w.r.t. other previous limit-cycles proposed in the literature. The PELC with the appropriate proposed reference frame linked to each sub-task achievement, permit to avoid efficiently and safely any obstacle shapes and to attract the vehicle toward any specific target in the environment. Thereafter, an optimal multi-criteria formulation of PELC* has been proposed to optimize the features of each elementary path (e.g., safety, curvilinear length, smoothness, etc.). To perform optimal long-term navigation, a multitude of PELC have been appropriately sequenced (while using the proposed Algorithm 1) to perform optimal global path planning based on PELC (gPELC*). Secondly, this paper made the focus on the generic, flexible and reliable proposed HHCA with its homogeneous set-points definition (based on PELC/PELC* or gPELC*) and common control law permitting to highly simplify: the use of HHCA for reactive and cognitive navigation and possible switch between them; the stability analysis of the overall multi-controller architecture. An extensive number of simulations, with several situations, and an actual vehicle experiment have been performed in order to confirm the large potentialities of the overall proposed methodology.

Future works aim to extensively apply the overall control architecture on actual autonomous vehicles. The formulation of the optimal balance between reactive and cognitive navigation is also among the most important issues to be addressed in near future.

Acknowledgments

This work was supported by LABEX IMobS3 (ANR-7107-LABX-716701) and the French National Research Agency (ANR) through the SafePlatoon (ANR-10-VPTT-0011) and R-Discover (ANR-08-CORD-0019) projects. The author would like to acknowledge strongly J. M. Vilca Ventura for his help concerning the experimental part.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.robot.2016.11.006.

References

[1] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics (Intelligent Robotics and Autonomous Agents), The MIT Press, 2005.
[2] H. Choset, K.M. Lynch, S. Hutchinson, G.A. Kantor, W. Burgard, L.E. Kavraki, S. Thrun, Principles of Robot Motion: Theory, Algorithms, and Implementations, MIT Press, Cambridge, MA, 2005.
[3] R. Siegwart, I. Nourbakhsh, D. Scaramuzza, Introduction to autonomous mobile robots, in: Intelligent Robotics and Autonomous Agents, MIT Press, 2011.
[4] S. Fleury, P. Souères, J.-P. Laumond, R. Chatila, Primitives for smoothing mobile robot trajectories, in: ICRA, 1993, pp. 832–839.
[5] J.-M. Vilca, L. Adouane, Y. Mezouar, A novel safe and flexible control strategy based on target reaching for the navigation of urban vehicles, Robot. Auton. Syst. 70 (2015) 215–226.
[6] S. Gulati, A framework for characterization and planning of safe, comfortable, and customizable motion of assistive mobile robots (Ph.D. thesis), The University of Texas at Austin, 2011.
[7] J. Minguez, F. Lamiraux, J.-P. Laumond, Handbook of Robotics, in: Motion Planning and Obstacle, Springer, 2008, pp. 827–852.
[8] M. Pivtoraiko, A. Kelly, Fast and feasible deliberative motion planner for dynamic environments, in: International Conference on Robotics and Automation, 2009.
[9] Y. Kanayama, Y. Kimura, F. Miyazaki, T. Noguchi, A stable tracking control method for an autonomous mobile robot, in: International Conference on Robotics and Automation, 1990.
[10] C. Samson, Control of chained systems. application to path following and point stabilization of mobile robots, IEEE Trans. Autom. Control 40 (1) (1995) 64–77.
[11] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Int. J. Robot. Res. 5 (1986) 90–99.
[12] F. Aurenhammer, Voronoi diagrams—a survey of a fundamental geometric data structure, ACM Comput. Surv. 23 (3) (1991) 345–405.
[13] J.-C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, Boston, MA, 1991.
[14] E. Rimon, D.E. Koditschek, Exact robot navigation using artficial potential flelds, IEEE Trans. Robot. Autom. 8 (5) (1992) 501–518.
[15] S.M. Lavalle, Rapidly-exploring random trees: A new tool for path planning, Tech. rep., Computer Science Dept., Iowa State University, 1998.
[16] R.J. Szczerba, P. Galkowski, I.S. Glickstein, N. Ternullo, Robust algorithm for real-time route planning, IEEE Trans. Aerosp. Electron. Syst. 36 (2000) 869–878.
[17] P. Morin, C. Samson, Control of nonholonomic mobile robots based on the transverse function approach, Trans. Robot. 25 (2009) 1058–1073.
[18] M. Mouad, L. Adouane, D. Khadraoui, P. Martinet, Mobile robot navigation and obstacles avoidance based on planning and re-planning algorithm, in: 10th International IFAC Symposium on Robot Control (SYROCO12), Dubrovnik, Croatia, 2012.
[19] Y. Koren, J. Borenstein, Potential field methods and their inherent limitations for mobile robot navigation, in: International Conference on Robotics and Automation, 1991, pp. 1398–1404.
[20] O. Brock, O. Khatib, High-speed navigation using the global dynamic window approach, in: ICRA, 1999, pp. 341–346.
[21] P. Ogren, N.E. Leonard, A convergent dynamic window approach to obstacle avoidance, IEEE Trans. Robot. 21 (2) (2005) 188–195.
[22] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, Int. J. Robot. Res. 30 (7) (2011) 846–894.
[23] T. Fraichard, Trajectory planning in a dynamic workspace: a state time approach, Adv. Robot. 13(1) (1999) 75–94.
[24] B. Jur-Van-Den, M. Overmars, Roadmap-based motion planning in dynamic Environments, IEEE Trans. Robot. 21(5) (2005) 885–897.
[25] J. Esposito, Conditional density growth (CDG) model: A simplified model of RRT coverage for kinematic systems, Robotica 31 (2013) 733–746.
[26] R.A. Brooks, A robust layered control system for a mobile robot, IEEE J. Robot. Autom. RA-2 (1986) 14–23.

[27] R.C. Arkin, Motor schema-based mobile robot navigation, Int. J. Robot. Res. 8 (4) (1989) 92–112.
[28] L. Adouane, N. Le Fort-Piat, Hybrid behavioral control architecture for the cooperation of minimalist mobile robots, in: International Conference on Robotics and Automation, New Orleans, USA, 2004, pp. 3735–3740.
[29] M. Egerstedt, X. Hu, A hybrid control approach to action coordination for mobile robots, Autom. 38 (1) (2002) 125–130.
[30] J. Toibero, R. Carelli, B. Kuchen, (2007) Switching control of mobile robots for autonomous navigation in unknown environments, in: IEEE International Conference on Robotics and Automation, pp. 1974–1979.
[31] L. Adouane, Hybrid and safe control architecture for mobile robot navigation, in: 9th Conference on Autonomous Robot Systems and Competitions, Portugal, 2009.
[32] M.A. Arbib, Perceptual structures and distributed motor control, in: V.B. Brooks (Ed.), Handbook of Physiology, Section 2: The Nervous System, II, Motor Control, Part 1, American Physiological Society, 1981, pp. 1449–1480.
[33] R. Zapata, A. Cacitti, P. Lepinay, DVZ-based collision avoidance control of non-holonomic mobile manipulators, JESA 38 (5) (2004) 559–588.
[34] R.C. Arkin, Behavior-Based Robotics, The MIT Press, 1998.
[35] C. Ordonez, E.G.C. Jr., M.F. Selekwa, D.D. Dunlap, The virtual wall approach to limit cycle avoidance for unmanned ground vehicles, Robot. Auton. Syst. 56 (8) (2008) 645–657.
[36] E. Gat, Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots, in: Proceedings of AAAI-92, San Jose, CA, 1992, pp. 809–815.
[37] R. Alami, R. Chatila, S. Fleury, M. Ghallab, F. Ingrand, An architecture for autonomy, Int. J. Robot. Res. 17 (1998) 315–337.
[38] A. Ranganathan, S. Koenig, A reactive robot architecture with planning on demand, in: Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, Vol. 2, 2003, pp. 1462–1468.
[39] P. Ridao, J. Batlle, J. Amat, G.N. Roberts, Recent trends in control architectures for autonomous underwater vehicles, Int. J. Syst. Sci. 30 (9) (1999) 1033–1056.
[40] C. Rouff, M. Hinchey, Experience from the DARPA Urban Challenge, Springer Publishing Company, Incorporated, 2011.
[41] R. Firby, An investigation into reactive planning in complex domains, in: Sixth National Conference on Artificial Intelligence, Seattle, 1987, pp. 202–206.
[42] E. Gat, Three-layer architecture, in: D. Kortenkamp, R.P. Bonnasso, R. Murphy, erd (Eds.), Artificial Intelligence and Mobile Robotics, AAAI Press, 1998, pp. 195–210.
[43] R. Arkin, Towards the unification of navigational planning and reactive control, in: AAAI Spring Symposium on Robot Navigation, 1989, pp. 1–5.
[44] K. Konolige, K. Myers, E. Ruspini, A. Saffiotti, The saphira architecture: A design for autonomy, J. Exp. Theor. Artif. Intell. 9 (1997) 215–235.
[45] D. Busquets, C. Sierra, R.L. de Mántaras, A multiagent approach to qualitative landmark-based navigation, Auton. Robots 15 (2) (2003) 129–154.
[46] H. C.-H. Hsu, A. Liu, A flexible architecture for navigation control of a mobile robot, IEEE Trans. Syst. Man Cybern. Part A 37 (3) (2007) 310–318.
[47] A. El Jalaoui, D. Andreu, B. Jouvencel, A control architecture for contextual tasks management: application to the auv taipan, in: Oceans 2005—Europe, Vol. 2, 2005, pp. 752–757.
[48] L. Adouane, Toward smooth and stable reactive mobile robot navigation using on-line control set-points, in: IEEE/RSJ, IROS'13, 5th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, Tokyo-Japan, 2013.
[49] A. Benzerrouk, L. Adouane, P. Martinet, N. Andreff, Multi Lyapunov function theorem applied to a mobile robot tracking a trajectory in presence of obstacles, in: European Conference on Mobile Robots (ECMR 2009), Milini/Dubrovnik Croatia, 2009.
[50] J.-M. Vilca, L. Adouane, Y. Mezouar, Robust on-line obstacle detection using data range for reactive navigation, in: SYROCO'12, Dubrovnik, Croatia, 2012.
[51] J.-M. Vilca, L. Adouane, Y. Mezouar, On-line obstacle detection using data range for reactive obstacle avoidance, in: IAS'12, Korea, 2012.
[52] J. Vilca, L. Adouane, Y. Mezouar, Reactive navigation of mobile robot using elliptic trajectories and effective on-line obstacle detection, Gyroscopy and Navigation 4 (1) (2013) 14–25.
[53] D.-H. Kim, J.-H. Kim, A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer, Robot. Auton. Syst. 42 (1) (2003) 17–30.
[54] M.S. Jie, J.H. Baek, Y.S. Hong, K.W. Lee, Real time obstacle avoidance for mobile robot using limit-cycle and vector field method, Knowl.-Based Intell. Inf. Eng. Syst. (2006) 866–873.
[55] L. Adouane, Orbital obstacle avoidance algorithm for reliable and on-line mobile robot navigation, in: 9th Conference on Autonomous Robot Systems and Competitions, Portugal, 2009.
[56] R.A. Soltan, H. Ashrafiuon, K.R. Muske, Ode-based obstacle avoidance and trajectory planning for unmanned surface vessels, Robotica 29 (2011) 691–703.
[57] L. Adouane, A. Benzerrouk, P. Martinet, Mobile robot navigation in cluttered environment using reactive elliptic trajectories, in: 18th IFAC World Congress, Italy, 2011.
[58] S.-M. Khansari-Zadeh, A. Billard, A dynamical system approach to realtime obstacle avoidance, Auton. Robots 32 (2012) 433–454. http://dx.doi.org/10.1007/s10514-012-9287-y.
[59] F.S. Segundo, J.R. Sendra, Degree formulae for offset curves, J. Pure Appl. Algebra 195 (3) (2005) 301–335.
[60] A. Cayley, Sur la courbe parallele a l'ellipse, in: The Collected Mathematical Papers, Vol. 4, Cambridge University Press, 2009, pp. 152–157. Cambridge Books Online. URL http://dx.doi.org/10.1017/CBO9780511703706.032.
[61] J. Denavit, R.S. Hartenberg, A kinematic notation for lower-pair mechanisms based on matrices, Trans. ASME, J. Appl. Mech. 22 (1955) 215–221.
[62] W. Khalil, E. Dombre, Modeling, Identification and Control of Robots, Hermes Penton, 2004.
[63] S. Gim, L. Adouane, S. Lee, J.-P. Derutin, Parametric continuous curvature trajectory for smooth steering of the car-like vehicle, in: 13th International Conference on Intelligent Autonomous System (IAS-13), Padova-Italy, 2014.
[64] F. Harary, Graph Theory, Addison-Wesley, Reading, MA, 1969.
[65] J. Bondy, U. Murty, Graph Theory, in: Graduate Texts in Mathematics, vol. 244, Springer, Berlin, 2008, xii, 651 p.
[66] E.W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1 (1959) 269–271.
[67] P. Pirjanian, Multiple objective behavior-based control, Robot. Auton. Syst. 31 (2000) 53–60. http://dx.doi.org/10.1016/S0921-8890(99)00081-0.
[68] L. Adouane, N. Le-Fort-Piat, Evolutionary parameters optimization for an hybrid control architecture of multicriteria tasks, in: International Conference on Robotics and Biomimetics (ROBIO), Shenyang-China, 2004.
[69] A. De Luca, G. Oriolo, C. Samson, Feedback control of a nonholonomic car-like robot, in: J.-P. Laumond (Ed.), Robot Motion Planning and Control, in: Lecture Notes in Control and Information Sciences, vol. 229, Springer, Berlin, Heidelberg, 1998, pp. 171–253.
[70] J.-M. Vilca, L. Adouane, Y. Mezouar, An adapted nominal control law for reactive navigation in cluttered environment for electric urban vehicle, in: 12th International Conference on Robotics and Automation, Germany, 2013.
[71] H.K. Khalil, Frequency domain analysis of feedback systems, in: Nonlinear Systems: Chapter 7, third ed., 2002.
[72] G.M. Siouris, Missile Guidance and Control Systems, Springer-Verlag, 2004.
[73] A. Benzerrouk, L. Adouane, P. Martinet, Stable navigation in formation for a multi-robot system based on a constrained virtual structure, Robot. Auton. Syst. 62 (12) (2014) 1806–1815.
[74] J.-M. Vilca, L. Adouane, Y. Mezouar, Adaptive leader-follower formation in cluttered environment using dynamic target reconfiguration, in: International Symposium on Distributed Autonomous Robotic Systems, DARS 2014, Daejeon-Korea, 2014.
[75] C. Goerzen, Z. Kong, B. Mettler, A survey of motion planning algorithms from the perspective of autonomous uav guidance, J. Intell. Robot. Syst. 57 (1–4) (2010) 65–100.
[76] P. Maes, The dynamics of action selection, in: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI), Detroit, 1989, pp. 991–997.
[77] M.J. Mataric, M. Nilsson, K. Simsarian, Cooperative multi-robots box-pushing, in: IEEE International Conference on Intelligent Robots and Systems, Vol. 3, 1995, pp. 556–561.
[78] M. Wang, J.N. Liua, Fuzzy logic-based real-time robot navigation in unknown environment with dead ends, Robot. Auton. Syst. 56 (7) (2008) 625–643.
[79] P. Pirjanian, Multiple objective behavior-based control, J. Robot. Auton. Syst. 31 (1) (2000) 53–60.
[80] S. IP.Data.Sets, March 2015. http://ipds.univ-bpclermont.fr.

Lounis Adouane is an Associate Professor since 2006 at the Institut Pascal-Polytech Clermont-Ferrand in France. He received an MS in 2001 from IRCCyN-ECN Nantes, where he worked on the control of legged mobile robotics. In 2005, he obtained a Ph.D. in automatic control from FEMTO-ST laboratory-UFC Besançon. During his Ph.D. studies he deeply investigated the field of multi-robot systems, especially those related to bottom-up and reactive control architectures. After that, he joined in 2005 Ampère laboratory-INSA Lyon and studied hybrid (continuous/discrete) control architectures applied to cooperative mobile robot arms. Dr. Adouane had the opportunity to visit several institutions/laboratories, such as 1 month in 2009 at LIST (Luxembourg) and 6 months in 2014 at Cranfield and Kingston universities (United Kingdom). In 2015, he obtained from Blaise Pascal University a HDR (habilitation to steer research in Robotics). Since 2006, he has authored/coauthored more than 70 international references and 2 books. His main research interests include: Autonomous mobile robots/vehicles, Behavioral/multi-controller architectures, Obstacle avoidance, Lyapunov-based synthesis and stability, Cooperative robotics, Task/trajectory planning and re-planning, Artificial intelligence, Multi-robot/agent simulation.