J Intell Robot Syst (2016) 82:301–324
DOI 10.1007/s10846-015-0223-1

Optimal Multi-Criteria Waypoint Selection for Autonomous Vehicle Navigation in Structured Environment

José Vilca · Lounis Adouane · Youcef Mezouar

Received: 17 November 2014 / Accepted: 25 February 2015 / Published online: 15 April 2015

Abstract This paper deals with autonomous navigation of unmanned ground vehicles (UGV). The UGV has to reach its assigned final configuration in a structured environments (e.g. a warehouse or an urban environment), and to avoid colliding neither with the route boundaries nor any obstructing obstacles. In this paper, vehicle planning/set-points definition is addressed. A new efficient and flexible methodology for vehicle navigation throughout optimal and discrete selected waypoints is proposed. Combining multi-criteria optimization and expanding tree allows safe, smooth and fast vehicle overall navigation. This navigation through way-points permits to avoid any path/trajectory planning which could be time consuming and complex, mainly in cluttered and dynamic environment. To evaluate the flexibility and the efficiency of the proposed methodology based on expanding tree (taking into account the vehicle model and uncertainties), an important part of this paper is dedicated to give an accurate comparison with another proposed optimization based on the commonly used grid map. A set of simulations, comparison with other methods and experiments, using an urban electric vehicle, are presented and demonstrate the reliability of our proposals.

Keywords Autonomous vehicle navigation · Optimal planning · Waypoints generation · Multi-criteria optimization · Vehicle's kinematic constraints · Localization under uncertainties

J. Vilca (✉) · L. Adouane · Y. Mezouar
Institut Pascal, Blaise Pascal University – UMR CNRS, 6602, Clermont-Ferrand, France
e-mail: Jose.Vilca@univ-bpclermont.fr

L. Adouane
e-mail: Lounis.Adouane@univ-bpclermont.fr

Y. Mezouar
e-mail: Youcef.Mezouar@univ-bpclermont.fr

1 Introduction

Fully autonomous vehicle navigation is a complex problem of major interest for the research community. Systems capable of performing efficient and robust autonomous navigation are unquestionably useful in many robotic applications such as manufacturing technologies [27], urban transportation [19], assistance to disabled or elderly people [25] and surveillance [30]. Even if a lot of progress has been made, some specific technologies have to be improved for real applications. This paper addresses specifically the problem of planning/set-points definition for autonomous navigation of vehicle in an urban environment (cf. Fig. 1).

1.1 Overview of Navigation Strategies

Different strategies for autonomous navigation have been proposed in the literature [2, 10, 22]. The most popular approaches are based on the tracking of a pre-defined reference trajectory [24]. The reference

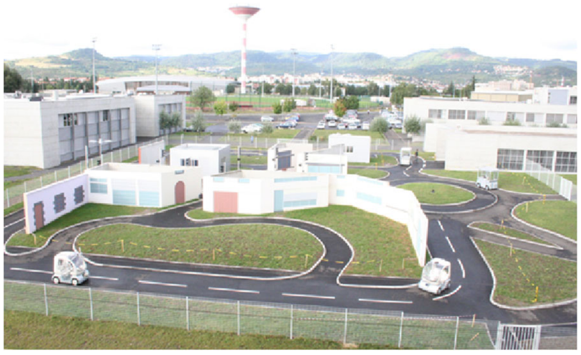


Fig. 1 Autonomous navigation of UGVs in a dedicated structured environment (Clermont-Ferrand, France)

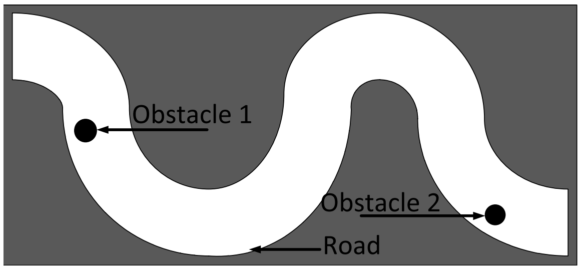
trajectory can be obtained using a combination of path planning and trajectory generation techniques [21]. The computation of a time-parameterized path while taking into account different vehicle constraints and environment characteristics is time-consuming [18] and [7]. Different algorithms for computation of a safe path (without temporal reference) require less computational time [21]. Typically, to obtain the reference path to be followed by the vehicle, arc-lines, B-splines or polynomial equations are used to interpolate points/waypoints [8, 11, 19] and [22]. A method to obtain these points was proposed in [28]. In [28], the authors use agent's observation and the geometric characteristics of the environment to select the waypoints. These waypoints can be used to reduce the planning time of existing planners. However, the method is based only on the position, the orientation and vehicle model are not taken into account. In [10], a feasible path is obtained using a polynomial curvature spiral. In [6], the trajectory generation method generates a smooth path considering the kinodynamic constraints of the vehicle. In [19], trajectories are built using user assigned points and interpolation functions such as cubic splines, trigonometric splines and clothoids. Moreover, velocity profiles along the trajectory are specified to improve the passengers comfort which is related to the acceleration. Nevertheless, trajectory generation presents some drawbacks such as the necessity of a specific planning method, a guarantee of continuity between different segments of the trajectory and more flexibility for dynamic replanning.

Otherwise, contrary to follow/track a trajectory to lead the robot toward its objective, few works in the literature propose to use only specific way-points in the environment to lead the robot toward its final objective. In [3], the authors propose a navigation via assigned static points for a unicycle robot. Nevertheless, the definition of the mission is less accurate because this strategy does not consider: the kinematic constraints of the robot (maximum velocity and steering), the orientation error and the velocity profile of the robot when it reaches the assigned point. In this paper, we present a navigation strategy which avoids the generation of a specific reference trajectory. Vehicle movements are generated using the control law that we have recently proposed in [33]. We will demonstrate in this paper that only few waypoints will be sufficient to guarantee safe and flexible vehicle navigation. We propose two methods to obtain these optimal waypoints in a known environment. These methods can take into account vehicle kinematic and perception/vehicle model uncertainties. The main advantage of this navigation strategy is its flexibility. The vehicle can perform different movements between waypoints without the necessity of replanning any reference trajectory, and it can also add or change the location of the successive waypoints according to the environment configuration or the task to achieve. This strategy allows thus flexible navigation while taking into account appropriate waypoints suitably placed in the environment.

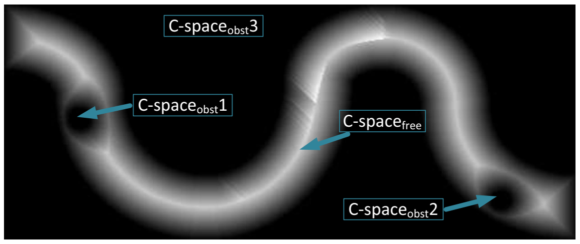
1.2 Related Works

Different algorithms can be used to obtain waypoints configuration such as A^*, D^* [7], Rapidly Random Tree (RRT) [18], Sparse A^* Search (SAS) [31]. At this aim, Configuration space (C-space), space of all possible configuration of the robot [29], enables the identification of the safe area where the vehicle can navigate without a collision risk (free space C-space$_{free}$). C-space is used to compute the minimum distance to C-space$_{obst}$ (obstacle or road boundaries space). Figure 2 shows the C-space and its Voronoï diagram [20] in gray scale w.r.t. the distance to the closest C-space$_{obst}$ (the whitest area represents the safest area).

Typically, algorithms based on grid map (e.g., A^* or D^*) produce the shortest path by optimization of a criterion such as the distance to the goal, distance to the risk area, etc. [7]. The algorithm begins generally at the final cell (final position) and traverses the cell's neighbors until to reach the initial position. The cost of traveling through the neighbor is added to the



(a)



(b)

Fig. 2 **a** Road scheme and **b** its C-space with its Voronoï diagram

total cost, the neighbor with the lowest total cost is selected, and so on. The process terminates once the initial position is reached. The path is given through the cell positions of the grid map while backtracking the cells which have the lower path cost, sometimes a polynomial interpolation is used to obtain a smooth path [8]. In [34], the authors present an A^* algorithm using clothoid trajectories assuming constant velocity along them. Therefore, appropriate waypoints can be selected from this shortest path while only considering the cells where an orientation change occurs (w.r.t. its predecessor). Nonetheless, this algorithm does not consider neither former/initial vehicle orientation nor its kinematic constraints.

Instead of using grid map, it is possible also to obtain safe, feasible and smooth path using expanding tree algorithms (e.g., RRT, RRT* or SAS [15, 18, 21] and [31]). This could be done by providing to the vehicle-model the commands to reach the successive selected nodes until the goal [18, 21] and [31]. The basic process of RRT consists in selecting, at each sample time, a random node q_{random} in the C-space$_{free}$. This selection considers generally only position $q_{random} = (x_{random}, y_{random})$ without any *a priori* vehicle orientation [21]. Then, the commands (discrete values) are applied to vehicle (from its current position and orientation) during a constant time t_{exp}. The vehicle model and constant commands allow to predict the final vehicle position at the end of t_{exp}. The commands that produce the closest position q_{chosen} (a node which optimizes a dedicated task criterion [32]) to current random node q_{random} are selected and stored with q_{choose}. A new expansion starts until to reach q_{random} or to select a new random node q^{new}_{random}. Therefore, the waypoints can be selected, as in the case of grid map, while only considering the nodes where an orientation change occurs (w.r.t. its predecessor node). Algorithms based on RRT are very useful for motion planning because they can provide the commands (based on the kinematic/dynamic model of the vehicle) to reach the desired final configuration [18] and [32]. Moreover, it avoids the use of grid maps that can increase the computation time for large environment. In [31], the authors use the expanding tree for trajectory planning introducing different constraints such as maximum turning angle and route distance. Nevertheless, this method does not consider neither the vehicle movements along the trajectory nor localization uncertainties. In [14], sequential composition of controllers (e.g., go to the landmark and wall following controller) are used to generate valid vehicle states q_{choose} to the navigation function. This approach avoids to find a single globally attractive control law and allows to use some additional sensing capability of the robot when the landmark is unreachable (e.g. GPS-denied area). However, the obtained navigation function has a complex computational processing. The most important drawbacks of expanding tree algorithms are the slow convergence to cover all space until to reach the goal and that in most cases it does not provide the shortest path since the nodes are randomly selected [1]. Furthermore, it is important to underline that in the RRT the control commands are maintained during a certain time, whereas in this paper the vehicle movement takes into account the definition of the used control law in addition to the vehicle model (cf. Section 3.2). A comparison with RRT and Voronoï approaches is shown in Section 4.1.2.

In this paper, we propose a method based on expanding tree to obtain the optimal waypoint configuration in a structured environment (cf. Fig. 4). It allows to consider constraints such as the kinematic model and the used control law. Criteria to optimize are related to the kinematic constraints of the vehicle (non-holonomy, maximum velocity and steering angle) and localization uncertainties. To highlight the advantages and flexibility of our proposal, a comparison with another proposed method, based on the commonly used grid map, is presented (cf. Section 3). The method based on grid map algorithm considers the vehicle as one cell without constraints and it can move only through the cells of the grid map. The trajectory of the vehicle depends on the choice of the waypoint configuration (cf. Section 4). We will show that the method based on grid map is less flexible and less efficient with regards to the methods based on expanding tree (cf. Section 4).

The rest of the paper is organized as follows: the next section presents the navigation framework, the waypoint assignment strategy for navigation, the vehicle model and its control law. The selection of waypoint configurations in the environment using a multi-criteria optimization techniques is described in Section 3. Simulation and experiments are given in Section 4. Finally, Section 5 provides a conclusion and prospects for future studies.

2 Overall Autonomous Navigation Framework

An important condition in the field of autonomous vehicle is to ensure safe and flexible vehicle navigation (cf. Fig. 4). In this work, safe navigation consists in not colliding with the road limits and other obstacles while respecting the physical constraints of the vehicle. Flexible navigation consists in allowing different possible movements to achieve the task, while guaranteeing a smooth trajectory of the vehicle. Certainly, the main idea of the proposed work is to guarantee both criteria simultaneously.

This paper focuses on the method to obtain the optimal set of waypoints appropriately located in the environment to perform a safe, flexible and fast vehicle navigation (cf. Section 3). Nevertheless, before detailing this multi-criteria optimization problem, let us first present in Section 2.1 the details of the autonomous navigation strategy based on finite and sequential waypoints assignment described in [33].

2.1 Navigation from Sequential Waypoint Assignment

A waypoint corresponds to a specific key configurations $(x_{q_i}, y_{q_i}, \theta_{q_i}, v_{q_i})$ (where $(x_{q_i}, y_{q_i})^T$, θ_{q_i} and v_{q_i} denote to respectively the position, the orientation and the velocity of the waypoint q_i) in the environment as given in Fig. 3 (cf. Section 3). A vehicle navigation using only waypoints allows to avoid any path/trajectory planning which could be time-consuming and complex, mainly in cluttered and dynamic environment. Moreover, this kind of navigation does not require the pose of the closest point to any trajectory (w.r.t. the robot configuration) and/or the value of the curvature at this point [10]. Consequently, the navigation problem is simplified to a waypoint reaching problem, i.e, the UGV is guided by waypoints (cf. Fig. 3) instead of following a specific fixed path. Moreover, it is important to notice that if the successive waypoints are closer to each other then the vehicle tends to perform a path following navigation. To simplify the computation of the waypoints, they could be selected from a pre-defined path if it is available [33]. Indeed, a safe reference path can be obtained by different algorithms such as a Voronoï diagram [20] or potential fields [17]. Nevertheless, adding this step of path planning (with all its possible drawbacks (cf. Section 4)) before obtaining the set of waypoints, restricts considerably the C-space$_{free}$ to only a curvilinear line. Thus, the optimality of the obtained set of Waypoints is not guaranteed (cf. Section 3).

The strategy proposed in [33] uses a sequence of waypoints suitably positioned in the environment. To navigate between successive waypoints (e.g. q_j and q_{j+1}), the distance of the vehicle to the target d and the error e_θ between the orientation of the vehicle and the target are used. Their maximum values (E_{dis} and $E_\angle$ respectively (cf. Fig. 3)) are imposed to the current waypoint q_j to be reached. These values are notably related to the inaccuracies of the vehicle localization and/or the performance of the used control law. The maximum authorized values allow to keep a reliable vehicle control towards the target T_j (cf. Fig. 3) while guaranteeing the appropriate vehicle configuration to reach the next target T_{j+1}.

Figure 3 shows a set of successive waypoints. D_j is the Euclidean distance between the waypoints q_{j-1} and q_j. For simplicity, the orientation of the waypoint θ_{q_j} is represented as the orientation of the line that joins q_j and q_{j+1}. The strategy to assign at each sample time the appropriate waypoint is shown in Algorithm 1. The parameters of the control law enable the vehicle to reach the next waypoint (cf. Section 2.3) while ensuring that the vehicle trajectory is always within the road boundaries (cf. Section 4). The error

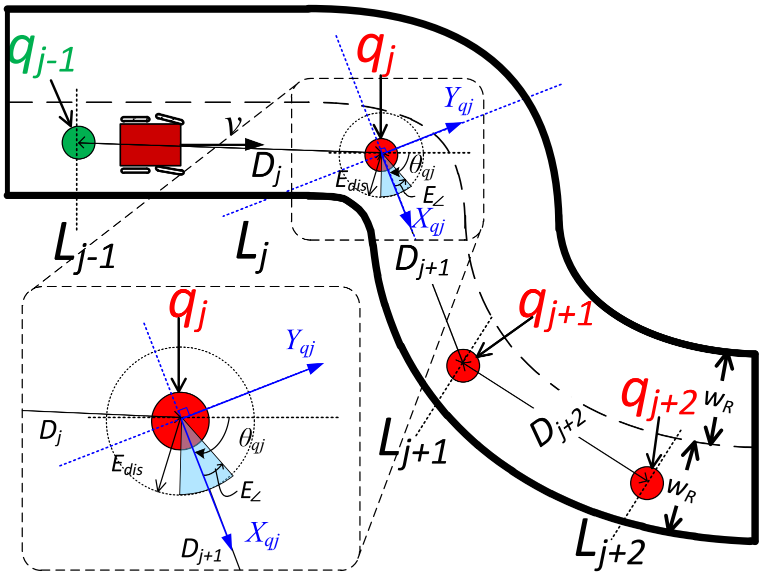


Fig. 3 Description of waypoints assignment

Algorithm 1 Sequential waypoint assignment

```
Require: Vehicle pose, current waypoint $q_j$ and set of waypoints
Ensure: Next waypoint $q_{j+1}$
1: if $((d \le E_{dis}$ and $e_\theta \le E_\angle))$ or $(x^{q_j} \ge 0)$ then
       ▷ $x^{q_j}$ is the coordinate of the vehicle in the local frame of the current waypoint $X_{q_j}Y_{q_j}$ (cf. Fig. 3)
2:     Switch from the current waypoint $q_j$ to the next sequential waypoint $q_{j+1}$
3: else
4:     Keep going to waypoint $q_j$
5: end if
```

conditions, E_{dis} and $E_\angle$, are used to switch to the next waypoint when the vehicle position enters a circle with a radius equal to E_{dis} and center (x_{q_j}, y_{q_j}). The current waypoint is updated with the following waypoint and the vehicle starts the movement to reach this new waypoint. If the vehicle does not satisfy the distance and orientation error conditions w.r.t. the current waypoint q_j then the perpendicular line L_j (Y_{q_j} axis) to the road at the current waypoint is used to switch to the next waypoint when the vehicle crosses L_j (cf. Fig. 3).

Before presenting briefly the control law to reach sequentially each single waypoint (cf. Subsection 2.3), let us present the navigation scenario and the model of the vehicle used for the control law definition.

2.2 Navigation Scenario and Vehicle Modeling

The following scenario is considered (cf. Figs. 2 and 4):

– The environment of navigation is known throughout a map, containing the position of all static obstacles.
– The kinematic/dynamic model of the vehicle is known (with potentials uncertainties).
– The vehicle starts at the initial pose P_i and it has to reach the final position P_f (in certain cases, $P_i = P_f$).

The UGV evolves in asphalt road and in cluttered urban environment with relatively low speed (less than

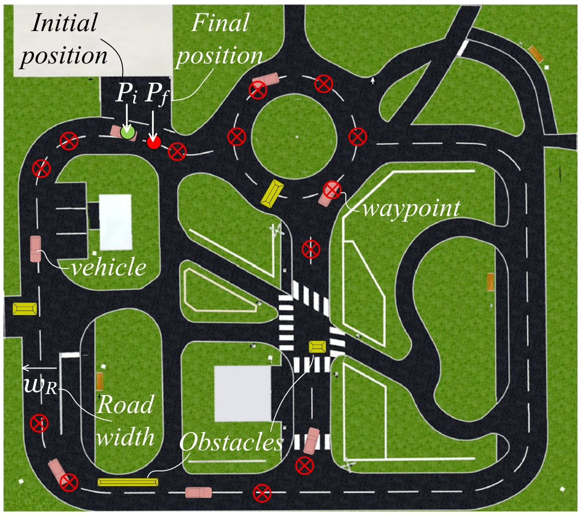


Fig. 4 Nominal scenario related to the effective platform (cf. Fig. 23) with the road map and the vehicle navigation

$v_{max} = 3\ m/s$). Hence, the use of kinematic model of the UGV is sufficient (which relies on pure rolling without slipping). The kinematic model of the UGV is based on the tricycle model [23]. The two front wheels are replaced by a single virtual wheel located at the center between the front wheels (cf. Fig. 5). In [13], different tricycle designs are described giving the relation between the vehicle wheels velocities and the global vehicle kinematic model (linear and angular velocities). This relation is important to take into account for dynamic modeling of the vehicle [13]. Nevertheless, as mentioned above, the kinematic model is enough for our application. The kinematic model is given by (cf. Fig. 5):

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = v\tan(\gamma)/l_b \end{cases} \tag{1}$$

where (x, y, θ) is the vehicle pose in the global reference frame $X_G Y_G$.v and γ are respectively the linear velocity and orientation of the vehicle front wheel. l_b is the wheelbase of the vehicle.

2.3 Target Reaching Controller

The target set-point modeling corresponds to a single waypoint configuration $(x_T,\ y_T,\ \theta_T,\ v_T)$, where $(x_T,\ y_T,\ \theta_T)$ and v_T are respectively the pose and velocity of the target. For static target reaching, $v_T \neq 0$ is considered as a desired velocity value for the vehicle when it reaches the desired target pose.

The target reaching controller guides the vehicle sequentially towards the current assigned static target (cf. Section 2.1). Before briefly describing the used control law [33], let us define the following notation (cf. Fig. 5):

- O_G and O_m are respectively the origin of global and local reference frames of the vehicle.
- I_{cc} is the instantaneous center of curvature of the vehicle trajectory, $r_c = l_b/\tan(\gamma)$ is the radius of curvature and $c_c = 1/r_c$ is the curvature.
- (e_x, e_y, e_θ) are the errors w.r.t local frame $(X_m Y_m)$ between the vehicle and the target poses.
- θ_{RT} and d are respectively the angle and distance between the target and vehicle positions.
- e_{RT} is the error related to the vehicle position (x, y) w.r.t the target orientation.
- d_l is the distance from the vehicle to the target orientation line.

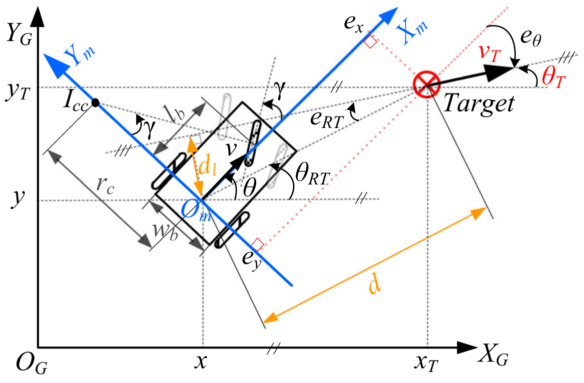


Fig. 5 UGV and target configuration variables in Cartesian references frames (local and global)

This controller is based on the pose control of the UGV w.r.t. the target. Let us first introduce the error variables (e_x, e_y, e_θ) (cf. Fig. 5) defined in the local reference frame $X_m Y_m$:

$$\begin{cases} e_x = \cos(\theta)(x_T - x) + \sin(\theta)(y_T - y) \\ e_y = -\sin(\theta)(x_T - x) + \cos(\theta)(y_T - y) \\ e_\theta = \theta_T - \theta \end{cases} \tag{2}$$

The error function e_{RT} is added to the canonical error system (2). The parameters d and θ_{RT} can be written as (cf. Fig. 5):

$$d = \sqrt{(x_T - x)^2 + (y_T - y)^2} \tag{3}$$

$$\begin{cases} \theta_{RT} = \arctan\left((y_T - y)/(x_T - x)\right) & \text{if } d > \xi \\ \theta_{RT} = \theta_T & \text{if } d \leq \xi \end{cases} \tag{4}$$

where ξ is a small positive value ($\xi \approx 0$). The error e_{RT} is defined as (cf. Fig. 5):

$$e_{RT} = \theta_T - \theta_{RT} \tag{5}$$

It can be written as a function of e_x, e_y and e_θ as:

$$\tan(e_{RT}) = \tan(e_\theta - (\theta_{RT} - \theta)) = \frac{e_x\tan(e_\theta) - e_y}{e_x + \tan(e_\theta)e_y} \tag{6}$$

where $\tan(\theta_{RT} - \theta) = e_y/e_x$ (cf. Fig. 5).

Finally the pose errors and velocities $(e_x, e_y, e_\theta, v_T)$ are the input of the control law. The control law is obtained from Lyapunov stability analysis (cf. Appendix). It guarantees that the static or dynamic target will be reached [33]. The desired vehicle linear velocity v and its front wheel orientation γ, that make the errors (e_x, e_y, e_θ) converge to zero, can be chosen as:

$$v = v_T \cos(e_\theta) + v_b \tag{7}$$
$$\gamma = \arctan(l_b c_c) \tag{8}$$

where v_b and c_c are given by:

$$v_b = K_x\,(K_d e_x + K_l d \sin(e_{RT}) \sin(e_\theta) + K_o \sin(e_\theta) c_c) \tag{9}$$

$$c_c = \frac{1}{r_{c_T}\cos(e_\theta)} + \frac{d^2 K_l \sin(e_{RT})\cos(e_{RT})}{r_{c_T} K_o \sin(e_\theta)\cos(e_\theta)} + K_\theta \tan(e_\theta) + \frac{K_d e_y - K_l d \sin(e_{RT})\cos(e_\theta)}{K_o \cos(e_\theta)} + \frac{K_{RT}\sin^2(e_{RT})}{\sin(e_\theta)\cos(e_\theta)} \tag{10}$$

$\mathbf{K} = (K_d, K_l, K_o, K_x, K_{RT}, K_\theta)$ is a vector of positive constants defined by the designer. K_d, K_l and K_o are respectively related to the desired convergence of the distance and lateral and angular errors w.r.t. the assigned target. K_x, K_{RT} and K_θ are related to the maximum linear and angular vehicle velocities (more details are given in [33]).

3 Optimal Multi-criteria Waypoint Selection (OMWS)

This section is dedicated to the selection of the discrete waypoints in structured environment (cf. Fig. 4) in order to perform safe and flexible vehicle navigation. The waypoints are obtained from an efficient and flexible methodology based on multi-criteria optimization using either grid map (cf. Section 3.1) or expanding tree (cf. Section 3.2).

In the both proposed OMWS (i.e., based on Grid Map (GM) and Expanding Tree (ET)), waypoints are selected considering safe position on the road, feasibility of trajectories (smooth changes between the successive points and vehicle constraints) and uncertainties.

The waypoints assignment strategies (cf. Sections 3.1 and 3.2) are formulated as an optimization problem and solved using dynamic programming [4] (cf. Formulation 1).

Formulation 1 (Optimization problem) *For each discrete state $x_k \in X$ where X is a nonempty and finite state space. The objective is to obtain the sequence of states to reach the final state x_K while minimizing the following cost function:*

$$C(x_K) = \sum_{k=1}^{K} g(Pred_{x_k} \rightarrow x_k) + h(x_K) \tag{11}$$

where $Pred_{x_k}$ is the predecessor state of x_k. g is the immediate traveling cost function to go from $Pred_{x_k}$ to x_k. h is the future traveling cost function (heuristic) to go from the current state to the final state x_K. When the current state is the final state x_K then $h(x_K)$ is equal to zero. This function h contributes to improve the convergence of the suboptimal solutions towards the global optimal one [5].

3.1 Optimal Multi-criteria Waypoint Selection Based on Grid Map (OMWS-GM)

Before describing the proposed algorithm and the criterion to optimize, let us provide some useful definitions. A grid map corresponds to a limited environment area decomposed generally on square cells [7] (cf. Figs. 6 and 7). Each cell of the grid map can be an obstacle or a free space (cf. Subsection 1.2 for the definition of C-space$_{obst}$ and C-space$_{free}$). The exterior limits of the C-space$_{free}$ area are defined by the user, even for open environment (cf. Fig. 4). For simplification, the dimension of the cells in the grid

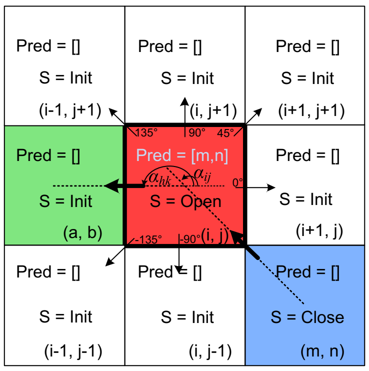


Fig. 6 A group of cells of the global grid map, the current cell ij (red), its predecessor cell (*blue*) and its probable successive cell (*green*). S is the cell state and $Pred$ is the predecessor of the cell

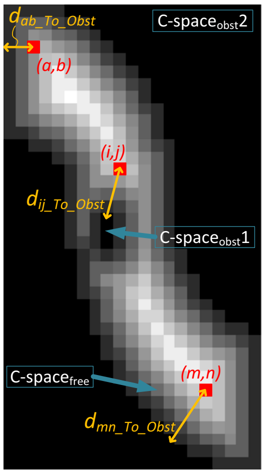


Fig. 7 Representation in gray scale w.r.t the distance to the closest C-space$_{\text{obst}}$ (the whitest area represents the safest area)

map is chosen according to the vehicle dimension. Therefore, the vehicle is contained, at each sample time, in only one cell [7]. We consider the center of the cell (i, j) as its position. Each cell (i, j) is defined by the following parameters:

- $\bar{w}_{ij} \in [0, 1]$ is related to the normalized distance $d_{ij\_To\_Obst}$ to the closest C-space$_{obst}$, and it is given by:

$$\bar{w}_{ij} = 1 - \frac{d_{ij\_To\_Obst}}{d_{max\_To\_Obst}} \tag{12}$$

  where $d_{max\_To\_Obst}$ is the maximum value among all $d_{cell\_To\_Obst}$ of all cells in the C-space$_{free}$. As an example, Fig. 7 shows different distances of cells localized at (a, b), (i, j) and (m, n) to the C-space$_{obst}$. $d_{max\_To\_Obst}$ is equal, in this example, to the maximum distance $d_{mn\_To\_Obst}$.
- S is the cell state, which has three possible values, $Init$ (Initialization), $Open$ (when it is in the expansion queue) and $Close$ (when it has already been expanded).
- A set of neighbors defined by:

$$\mathbf{S}_N(cell_{ij}) = \{(i \pm \Delta i_N, j \pm \Delta j_N) | (\Delta i_N, \Delta j_N) \neq (0, 0)\} \tag{13}$$

  where $\Delta i_N, \Delta j_N = 1 \ldots, N_h$. N_h is the neighborhood value (cf. Fig. 8). Figure 8a shows the case where $N_h = 1$ (which implies 8 neighbor cells). Figure 8b shows a larger neighbor cells when $N_h = 2$ (24 neighbors).
- $Pred_{ij}$ is the neighbor cell of ij which minimizes the total cost $C(ij)$ (cf. Fig. 6).
- $g(Pred_{ij} \rightarrow ij)$ is the traveling cost from the predecessor cell until the current cell ij.
- $h(ij)$ is the heuristic traveling cost from the current cell ij to the final cell. As conventional, this cost depends on the euclidean distance from the cell ij to the final cell.

The traveling cost function $g(Pred_{ij} \rightarrow ij) = g(mn \rightarrow ij)$, from mn to ij, is normalized ($g \in [0, 1]$). It is also designed to take into account the variation of cell orientation (cf. Fig. 6). It allows to obtain an optimal path consisting of minimum number of straight lines. Therefore, a lowest possible number of waypoints in the safe area can be extracted from this optimal path. The cost function $g(mn \rightarrow ij)$ is given by:

$$g(mn \rightarrow ij) = k_g \bar{w}_{ij} + (1 - k_g) \frac{|\alpha_{ij} - \alpha_{mn}|}{2\pi} \tag{14}$$

where the first term of Eq. 14 is related to the safety of the obtained solution, and the real constant $k_g \in [0, 1]$ is used to increase or to decrease the significance of this term. The second term of Eq. 14 is related to the smoothness of the obtained solution, i.e, the path has a limited and minimum orientation change. The cell orientations α_{mn}, $\alpha_{ij} \in]-\pi, \pi]$ are computed using the position of the current cell (i, j), its predecessor (m, n) and its probable successor (a, b) (cf. Fig. 6). They are computed as:

$$\alpha_{ij} = \arctan\left([a - i]/[b - j]\right) \tag{15}$$

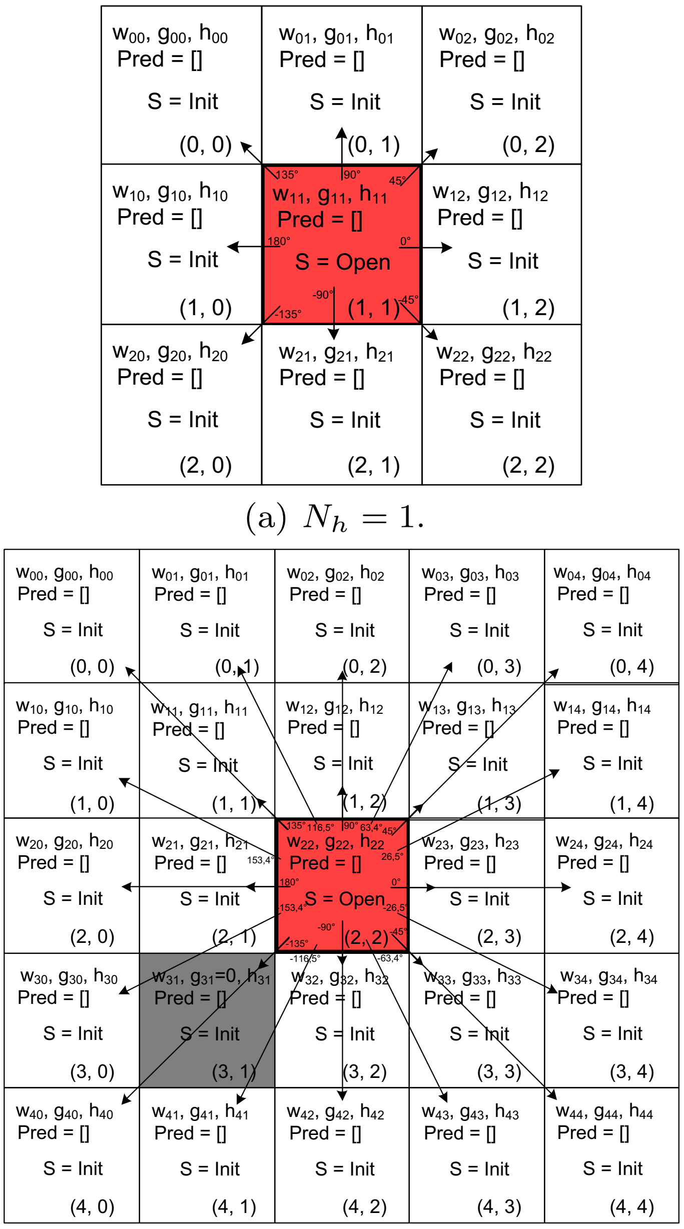
$$\alpha_{mn} = \arctan\left([i - m]/j - n]\right) \tag{16}$$

The heuristic traveling cost $h(ij) \in [0, 1]$ (refers to Eq. 11) is designed in function of the euclidean distance d_{ij} from the cell ij to the final cell. It is also used for the OMWS-ET (cf. Section 3.2). The cost function $h(ij)$ is given by:

$$h(ij) = k_h \left(1 - e^{-d_{ij}/k_e}\right) \tag{17}$$

where $k_h \in [0, 1]$ allows to tune the significance of $h(ij)$ in the total cost function (11). The exponential function was chosen because it gives values between 0

Fig. 8 Different neighborhood values N_h of the current cell (red). A gray cell represents a cell where the movements are forbidden



(a) $N_h = 1$.

(b) $N_h = 2$.

and 1 for positive values of d_{ij}. The constant $k_e \in \mathbb{R}^+$ is used to scale the value of d_{ij} according to the dimensions of the environment. The value of $h(ij)$ Eq. 17 decreases while the next selected cell goes closer to the final cell.

Algorithm 2 shows in pseudocode, the first proposed method to obtain the set of waypoints in a structured and cluttered environment. It starts from the final vehicle position (initial cell). The algorithm selects the cells that have the lower total cost $C(ij)$ (11) until to reach the final cell. The set of waypoints is finally obtained, while tracking the predecessor cell of each selected cell which minimizes the total cost.

Algorithm 2 Waypoint selection based on a grid map

```
Require: Initial position $p_i$, final position $p_f$ and a $Gridmap$
Ensure: Set of waypoints $S_p$
1: Init $state_{ij} = INIT$, $g_{ij} = 0$ and $Pred_{ij} = \oslash$, $\forall\ ij \in Gridmap$
2: Init $cell_{ij} = p_f$ and the set of neighbors $S_N(cell_{ij})$
3: while $cell_{ij} \neq p_i$ do ▷ Until to reach the initial position
4:     Set $state_{ij} = CLOSE$
5:     for $cell_N \in \mathbf{S}_N(cell_{ij})$ do
6:         if $\bar{w}_{cell_N} \neq 0$ then ▷ Only cells in the free space
7:             Obtain $cell_{pred} = Predecessor(cell_{ij})$
8:                                         ▷ When $cell_{ij} \neq p_f$
9:             Compute $\alpha_{ij}$ (15) and $\alpha_{mn}$ (16)
10:                        ▷ These values are 0 when $cell_{ij} = p_f$
11:            Compute the total cost $C(cell_N)$ (11)
12:            if $state_{cell_N} == INIT$ then
13:                $Pred_{cell_N} = cell_{ij}$ and add to the queue $Q$
14:                Set $state_{cell_N} = OPEN$
15:            else if $state_{cell_N} == OPEN$ then
16:                Update the queue $Q$
17:                $Pred_{cell_N}$ is the cell with lower total cost
18:            end if
19:        end if
20:    end for
21:    Sort the queue $Q$ in ascending order of total cost $C$
22:    Get the first value of queue $cell_{ij} = Q(first)$ and remove it from $Q$
23: end while
24: $S_p$ is the set of predecessor cells of $cell_{ij} = p_i$.
```

The obtained path is defined by straight lines connecting each two consecutive waypoints which belong to the set of obtained waypoints (cf. Algorithm 2). The smoothness of the path depends on the number of possible neighbors of the expanded cell defined by N_h (cf. Fig. 9). The drawback of using a large number of neighbors is obviously the increasing of processing time. When $N_h > 1$, it is mandatory to check if the current neighbor is blocked by some other forbidden neighbor (cf. Fig. 8). For an off-line planning, $N_h > 1$ can always be used to obtain a coherent and optimal solution regardless of time consumption.

3.2 Optimal Multi-criteria Waypoint Selection Based on Expanding Tree (OMWS-ET)

This subsection presents in details the main contribution of this paper for the optimal planning of the vehicle path, using an appropriate expanding tree. The formulation of this expanding tree integrates the kinematic model of the vehicle as well as the used control law definition and the vehicle localization uncertainties.

Before describing the proposed method and the criterion to be optimized, let us present the definition of expanding tree. The expanding tree is co mposed by nodes and edges which have the following properties:

- Each node q_j is defined by its pose $(x_{q_j}, y_{q_j}, \theta_{q_j})^T$, one predecessor node q_i (except for the initial node) and a traveling cost values $g(q_j)$ and $h(q_j)$ (cf. (11)).
- Each edge ξ_{ij} corresponds to the link between q_i to q_j nodes.
- $g(q_i \rightarrow q_j) = g(\xi_{ij})$ is the traveling cost from q_i to q_j.
- $h(q_j)$ is the heuristic traveling cost from the current node q_j to the final node (final vehicle pose). It is defined by Eq. 17 (cf. Section 3.1).

The traveling cost $g(\xi_{ij}) \in [0, 1]$ is designed to obtain an appropriate balance among safe, smooth, feasible and fast trajectory of the vehicle. It is defined as:

$$g(\xi_{ij}) = k_1\bar{w}_j + k_2\Delta\bar{v}_{ij} + k_3\Delta\bar{\gamma}_{ij} + k_4\Delta\bar{e}_{l_{ij}} \tag{18}$$

where k_1, k_2, k_3 and $k_4 \in \mathbb{R}^+$ are constants which are defined by the designer to give the right balance (according to context of navigation, e.g., focus more on the safety with regard to the smoothness) of each term of the criterion (18). To normalize the traveling cost, $k_i | i = 1, \ldots, 4$ must satisfy:

$$k_1 + k_2 + k_3 + k_4 = 1 \tag{19}$$

The normalization of the individual criterion given in Eq. 18 allows to simplify the choice of k_i to select the priority of a term w.r.t. the others according to the navigation context. In Section 4, different set of values k_1, k_2, k_3 and k_4 will be considered for different scenarios.

The first term of the cost function (18) is related to the safety of the navigation (12). The second and third terms are respectively related to the speed (20) and smoothness (23) of the trajectory. The fourth term is related to feasibility of the vehicle trajectory while considering localization uncertainties, i.e., the risk to collide with an obstacle while considering inaccuracies in the vehicle position and orientation (a detailed explanation of this term is given later in this subsection). This last term allows to consider the kinematic model of the vehicle and the control law. The details of each term is given in the following:

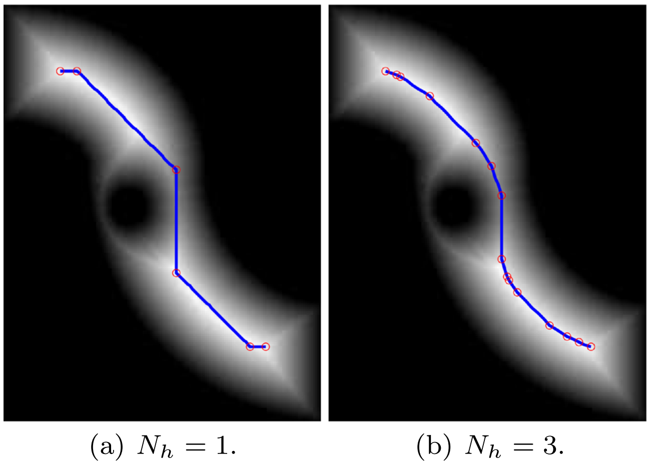


Fig. 9 Different sets of waypoints for different number of possible neighbor cells

– The term $\bar{w}_j \in [0, 1]$ is related to the distance from the node q_j to the closest C-space$_{obst}$. It is given by Eq. 12 (cf. Section 3.1).
– The term $\Delta\bar{v}_{ij} \in [0, 1]$ is related to the velocity from q_i to q_j, v_{ij}. It is given by:

$$\Delta\bar{v}_{ij} = 1 - \frac{v_{ij}}{v_{max}} \tag{20}$$

where v_{max} is the maximum velocity of the vehicle. We estimate v_{ij} as a function of the curvature of the trajectory. The maximum velocity occurs when the curvature is zero (straight line) and the minimum velocity $v_{min} \neq 0$ occurs when the curvature is bigger than the value corresponding to γ_{max} (cf. Fig. 5). This consideration allows the vehicle maneuvers without risk of collisions while enhancing the passenger comfort [19] (indeed, this permits to limit the centripetal forces). The minimum and maximum values of velocity and steering angle are defined by the designer according to the vehicle characteristics. The curvature is estimated using the orientation of the current node and its predecessor. Therefore, v_{ij} is computed as:

$$v_{ij} = v_{max} - \Delta\bar{\theta}_{ij}(v_{max} - v_{min}) \tag{21}$$

where $\Delta\bar{\theta}_{ij} \in [0, 1]$ is the normalized estimated curvature related to the variation of orientation between the current node q_j and its predecessor q_i. It is defined as:

$$\Delta\bar{\theta}_{ij} = \frac{|\theta_j - \theta_i|}{\Delta\theta_{max}} \tag{22}$$

where $\Delta\theta_{max}$ is the maximum variation between a probable orientation of the current node w.r.t

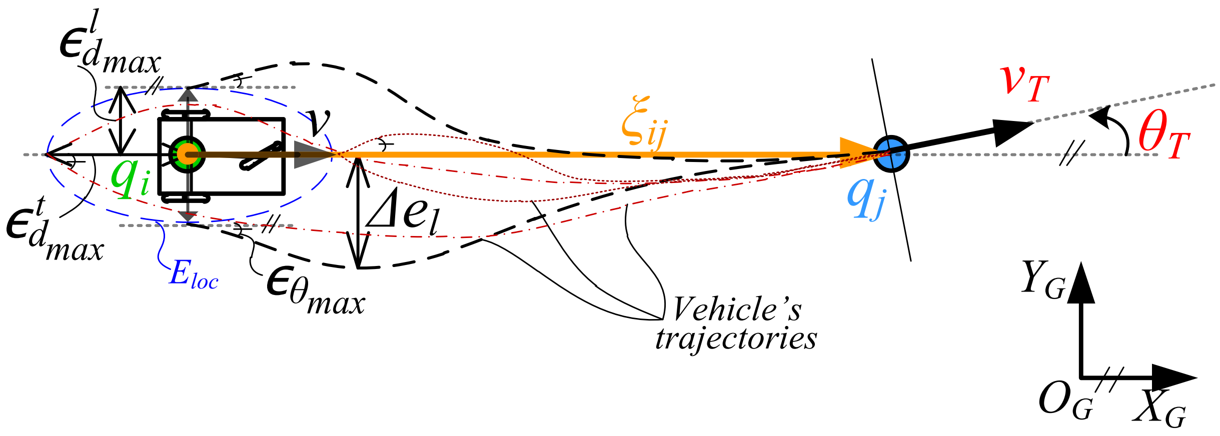


Fig. 10 Vehicle's trajectories which starts from extreme configurations ($\pm\epsilon^l_{d_{max}}$, $\pm\epsilon^t_{d_{max}}$ and $\pm\epsilon_{\theta_{max}}$) in the localization uncertainties ellipse E_{loc}.Δe_l is the maximum lateral deviation of all vehicle trajectories

the orientation of its predecessor. This value is defined according to the steering capability of the vehicle. θ_j and θ_i are computed using the node positions and Eqs. 15 and 16 (cf. Section 3.1).

- The term $\Delta\bar{\gamma}_{ij} \in [0, 1]$ is related to the variation of steering angle along the vehicle trajectory from q_i to q_j (for instance, Fig. 10 shows a vehicle trajectory between two nodes). It is given by:

$$\Delta\bar{\gamma}_{ij} = \frac{\sum_{q_i}^{q_j} |\Delta\gamma_{ij}|}{n_{q_{ij}}\gamma_{max}} \tag{23}$$

where $n_{q_{ij}}$ is the considered point number of the vehicle trajectory between q_i and q_j, and γ_{max} is the maximum steering angle of the vehicle. This term $\Delta\bar{\gamma}_{ij}$ (23) computes the sum of the $\Delta\gamma_{ij}$ to obtain the total variation of the steering angle along the vehicle trajectory. $\Delta\bar{\gamma}_{ij}$ uses the kinematic model and the control law to estimate the vehicle trajectory and commands or control set-points.

- The term $\Delta\bar{e}_{l_{ij}} \in [0, 1]$ is the normalized maximum deviation of vehicle trajectory w.r.t. the straight line that joins the positions (x_q, y_q) of q_i and q_j (cf. Fig. 10). It is computed as:

$$\Delta\bar{e}_{l_i j} = \frac{|\Delta e_{l_{ij}}|}{max\{\Delta e_l\}} \tag{24}$$

where $max\{\Delta e_l\}$ is the maximum deviation of all trajectories from the node q_i to the node q_j while considering the position and orientation uncertainties (ϵ_d and ϵ_θ respectively given in Fig. 10). This term $\Delta\bar{e}_{l_i j}$ allows to estimate the collision risk using the vehicle trajectory that takes into account the kinematic model, the control law and localization uncertainties (position and orientation). Figure 10 shows an illustration where the vehicle has an ellipse of localization uncertainties with axes ϵ_d^l and ϵ_d^t. The vehicle trajectories start at $\pm\epsilon_d^l$ in lateral distance (longitudinal distance is set to 0), and $\pm\epsilon_d^t$ in longitudinal distance (lateral distance is set to 0) from the vehicle position with a $\pm\epsilon_\theta$ from the vehicle presumed orientation, i.e., we consider all extreme configurations to obtain, according to these maximum error configurations, the maximum lateral deviation (Δe_l). The trajectories are obtained using the kinematic model and the used control law in an offline simulated procedure.

Algorithm 3 shows in pseudocode, the proposed method which uses expanding tree to obtain the optimal waypoints configurations w.r.t. the optimized multi-criteria function (18). Figure 11 shows the first steps of the algorithm where, for instance, the branch numbers of each node is $n_t = 3$ and each branch orientation w.r.t. the vehicle orientation is given by:

$$\alpha = \pm i\Delta\alpha,\ i = \begin{cases} 0, 1, \ldots, (n_t - 1)/2; & \text{if } n_t \text{ is odd} \\ 1, 2, \ldots, n_t/2; & \text{if } n_t \text{ is even} \end{cases} \tag{25}$$

where $\Delta\alpha$ is a constant angle defined according to the vehicle characteristics.

Algorithm 3 Waypoint selection based on expanding tree

```
Require: Initial pose $p_i$, final pose $p_f$, branch number $n_t$,
    edge distance $\xi$, branch orientation $\Delta\alpha$, tolerable error
    distance $\epsilon$ and C-space$_{free}$
Ensure: Set of waypoints $S_p$
 1: Init the initial node $q_0 = p_i$, $g_0 = 0$ and $Pred_{q_0} = \emptyset$
 2: Init the current node to expand $q_i = q_0$
 3: Init $Tree(q_i)$ =Expansion_Tree (Procedure 4) with $\alpha = 0$
    ▷ Initial expansion
 4: Set the new node to expand $q_i = r_t$ where $r_t \in Tree(q_i)$
 5: Set $Pred_{r_t} = q_i$ and compute the total cost $C(r_t)$ (11)
 6: while $|q_i - p_f| < \epsilon$ do
 7:     Compute the $Tree(q_i)$ = Expansion_Tree
 8:         ▷ refers to Procedure 4 with the set of $\alpha = \pm i\Delta\alpha$
 9:     for $r_t \in Tree(q_i)$ do
10:         if $r_t \in$ C-space$_{free}$ then
11:             Compute the total cost $C(r_t)$(11)
12:             $Pred_{r_t} = q_i$
13:             Add $r_t$ to the queue $Q$
14:         end if
15:     end for
16:     Sort the queue $Q$ in ascending order of total cost $C$
17:     Get the first value of queue $q_i = Q(first)$ and remove
    it from $Q$
18: end while
19: $S_p$ is the set of predecessor nodes of $q_i = p_f$.
```

Procedure 4 Expansion_Tree

```
Require: Current node $q_i$, set of $\alpha$ $S(\alpha)$, edge distance $\xi$
Ensure: Nodes of $Tree(q_i)$
 1: Init $Tree(q_i) = \emptyset$
 2: for $\alpha_t \in S(\alpha)$ do
 3:     Compute the orientation $\theta_{r_t} = \theta_{q_i} + \alpha_t$
 4:     Compute pose $r_t = q_i + [\xi\cos(\theta_{r_t}), \xi\sin(\theta_{r_t}), \alpha_t]^T$
 5:     Add $r_t$ to $Tree(q_i)$
 6: end for
```

The edge distance ξ is the Euclidean distance between two successive nodes and it depends on the environment dimensions, e.g., if the environment has a narrow passage then ξ must cope with this dimension. We consider that the edge orientation is the vehicle orientation at the current node position (cf. Fig. 11). Thus, at beginning the first expansion of q_0 is given with $\alpha = 0$ because the vehicle starts at initial fixed

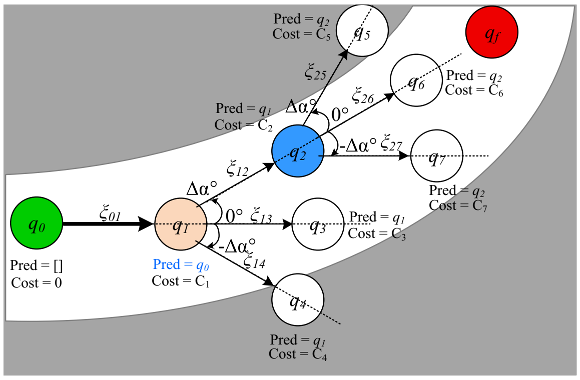
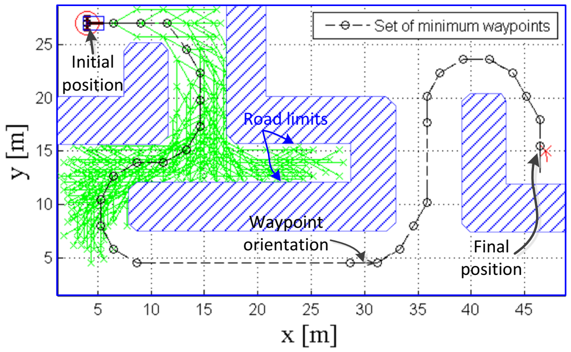
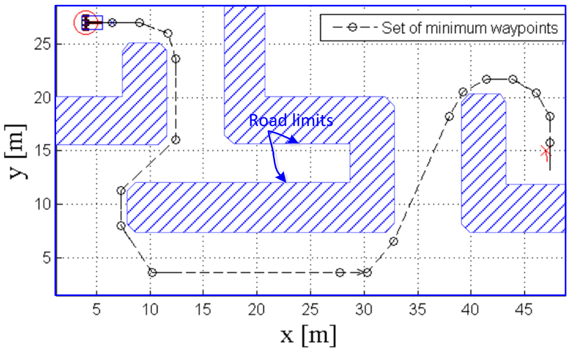


Fig. 11 Expanding tree method to obtain the appropriate set of waypoints



(a) Safest path: $k_1 = 1.0$, $k_2 = k_3 = k_4 = 0.0$ and $k_h = 0.1$



(b) Shortest path: $k_2 = 1.0$, $k_1 = k_3 = k_4 = 0.0$ and $k_h = 0.1$

Fig. 12 Set of waypoints for different parameters values k_i of the traveling cost

pose (cf. line 3 − 5 of Algorithm 3). This initial expansion is made to respect the kinematic constraints where the rotation of the vehicle requires a displacement (linear velocity $\neq$ 0) of the vehicle. Therefore, the successive node q_1 has different possible orientation and so on (cf. Fig. 11). The algorithm selects the node which has the lower total cost $C(q_j)$ (11). When two or more nodes have the same cost, the algorithm selects the last saved node. Figure 11 shows the successive steps, the node q_2 was selected from the expansion of q_1 $\{q_2, q_3, q_4\}$, which has the lower total cost value. The set of waypoints is obtained while tracking the predecessor nodes of the nodes with lower total cost. The selection of the node with lower total cost (cf. Algorithm 3, line 16 − 17) allows to avoid the deadlock areas because the successive branches from the nodes in this deadlock area will be in C-space$_{obst}$ (cf. Fig. 12a).

The smoothness of the vehicle trajectory depends on number of branches of each tree n_t, maximum branch orientation $\alpha_{max} = n_t\Delta\alpha/2$ and edge distance ξ (cf. Section 4). The drawback of using a large number of n_t is the increasing of the processing time required to obtain the set of waypoints. The vertex distance ξ is set to detect obstacles between the successive nodes.

This method uses deterministic selection of expanding tree to obtain the optimal solution with lowest total cost. Nevertheless, a feasible solution can be obtained using a probabilistic selection of expanding tree to decrease the processing time (cf. Section 4), i.e., the branch orientation α and edge distance ξ are randomly selected in a fixed interval [21]. In simulation, we will show in Section 4.1 the case where these parameters are randomly chosen.

As described above, the traveling cost (18) depends on four parameters ($k_i | i = 1, \ldots, 4$, which satisfy (19)) related respectively to the safety, velocity, less steering and taking into account uncertainties. The values of these parameters are fixed according to the desired navigation and environment conditions. A pragmatical procedure to set these parameters consists in first identifying the main desired vehicle behavior and setting its parameter k_i with a value greater than 0.5 (cf. Fig. 12). The other parameters will be tuned according to the designers secondary priorities. Figure 12 shows the set of waypoints when only the term with highest priority is considered in the traveling cost function. For instance, in Fig. 12a and b the priority is given respectively to the safest and the shortest paths. More examples of different tuned parameters will be shown in Section 4.1.

3.3 Minimum Set of Waypoints

Algorithm 2 and 3 were applied to obtain a set of waypoints S_p characterized by $(x_{q_j}, y_{q_j}, \theta_{q_j}, v_{q_j})$ on a specific environment. The proposed Algorithm 5 will allow to reduce the number of waypoints. Basically, this algorithm keeps only the waypoints (its pose, velocity and predecessor are stored) which have an orientation changes w.r.t. its predecessors.

Algorithm 5 Minimum set of waypoints

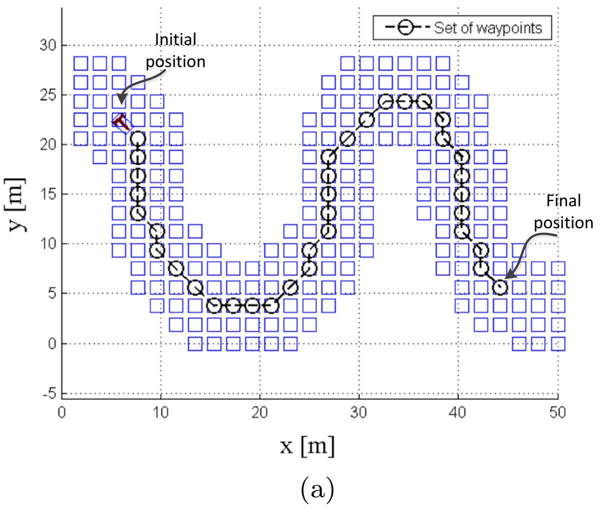
```
Require: Set of waypoints $S_p$ and $\Delta\theta_{max} \in \mathbb{R}^+$
Ensure: Minimum set of waypoints $S_pmin$
 1: Init $S_pmin = \{q_0\}$
 2: for $q_i \in S_p$ with $i > 1$ do ▷ Current waypoint compares its orientation w.r.t. predecessor waypoint
 3:     Compute $\Delta\theta = |\theta_{q_i} - \theta_{q_{i-1}}|$
 4:     if $\Delta\theta \geq \Delta\theta_{max}$ then
 5:         ▷ Check if predecessor waypoint belongs to $S_pmin$
 6:         if $q_{i-1} \notin S_pmin$ then
 7:             Add predecessor waypoint $q_{i-1}$ to $S_pmin$
 8:         end if
 9:         Add current waypoint $q_i$ to $S_pmin$
10:     end if
11: end for
```

4 Validation

This section presents a set of experiments to demonstrate the efficiency of our methods for autonomous navigation in a structured environment. Section 4.1 provides different scenarios to show the validity of our proposals. Section 4.2 discusses experimental results applied to an urban electric vehicle.

4.1 Simulations Results

This section shows optimal sets of waypoints, obtained according to the environment characteristics and/or the task to achieve. In what follows, it will be shown: a comparison between grid map and expanding tree algorithms (cf. Section 4.1.1); a comparison between the proposed OMWS-ET and a variation of RRT (cf. Section 4.1.2); different specific scenarios such as trajectory generation (cf. Section 4.1.3); a comparison between deterministic and probabilistic waypoints selection (cf. Section 4.1.4); an application of the proposed methodology of waypoints selection for multi-robot formation (cf. Section 4.1.5) and finally, Section 4.1.6 shows the flexibility of our proposal for local replanning of the waypoints configurations when unexpected obstacles are detected. For these simulations, the physical parameters of the UGV are based on the urban vehicle VIPALAB (cf. Fig. 22) which is modeled as a tricycle (1). Its dimensions are 1.27 m (width), 1.96 m (length) and 2.11 m (height). The UGV constraints are minimum movement velocity $v_{min} = 0.1\ m/s$, maximum velocity 1.5 m/s, maximum steering angle $\gamma_{max} = \pm 30°$ and maximum linear acceleration 1.0 m/s^2. The controller parameters are set to $\mathbf{K} = (1, 2.2, 8, 0.1, 0.01, 0.6)$ (cf. Section 2.3). These parameters were chosen to obtain



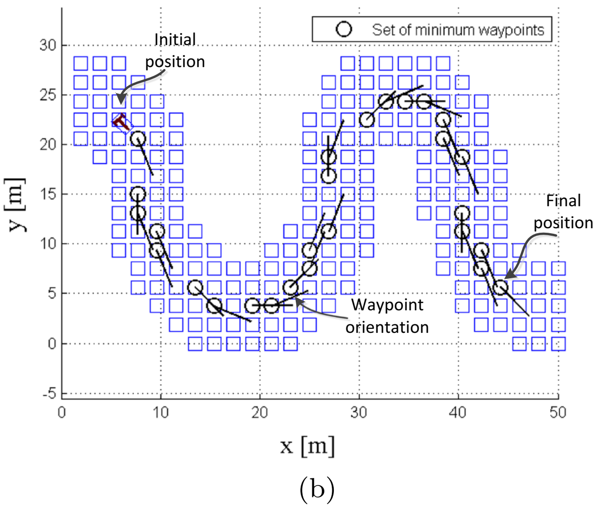


Fig. 13 **a**) Set of obtained waypoints using Algorithm 2 based on grid map and **b**) Minimum set of waypoints obtained by Algorithm 5

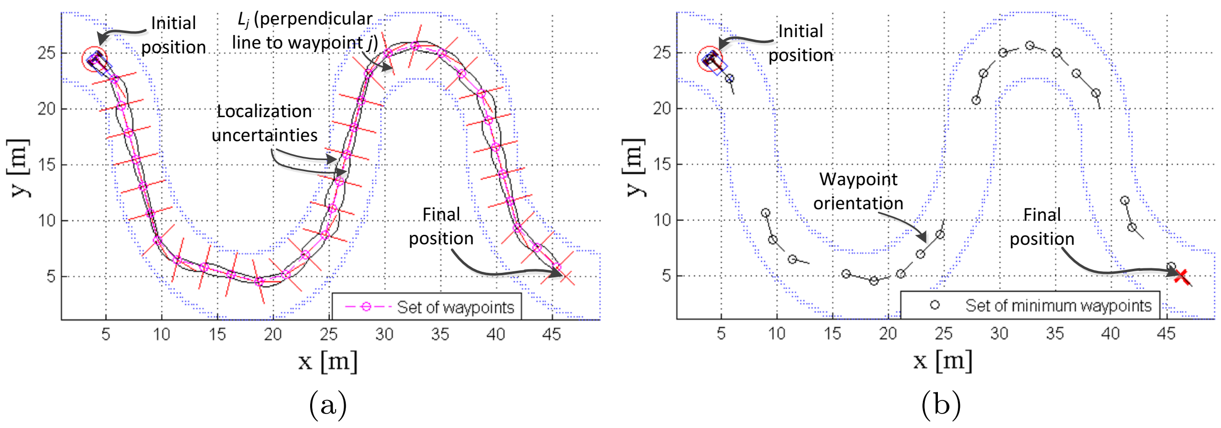


Fig. 14 **a**) Set of obtained waypoints using Algorithm 3 based on expanding tree and **b**) Minimum set of waypoints obtained by Algorithm 5

a good balance between: accurate and fast response and smooth trajectory while taking into account the limits of the vehicle structural capacities. We consider that the sample time is 0.01 s and a maximum number of iteration is $n_I = 5000$ to stop both algorithms, OMWS-GM (Algorithm 2) and OMWS-ET (Algorithm 3), when none solution can be obtained.

4.1.1 Grid Map versus Expanding Tree

These simulations show two set of waypoints obtained by the two proposed methods based on grid map and expanding tree (cf. Algorithms 2 and 3 respectively). Figures 13a and 14a show the set of obtained waypoints according to Algorithm 2 and 3. The minimum set of waypoints, obtained according to Algorithm 5, are given afterward in Figs. 13b and 14b.

For the grid map case, the cell has the vehicle dimension (2 m) and its neighborhood is $N_h = 1$. The constant value of k_g is 0.6 (14) and k_h is 0.1 (17). The minimum set of obtained waypoints has $n_w = 27$ elements. An additional constraint is considered for OMWS-GM (before line 11 of the Algorithm 2), the angle variation (second term of Eq. 14 must be less than a threshold θ_{th}). This constraint enables the processing time of the algorithm to be reduced since it considers only the cells with an orientation change, w.r.t the last cell orientation, less than θ_{th} (cf. Fig. 6).

For Expanding Tree case, the branch number n_t is 5, the edge distance ξ is 2.5 m and $\Delta\alpha$ is 15°. We consider the safety gain k_1 (cf. (18)) as the highest priority in this simulation. The constant values of $k_i|i = 1, \ldots, 4$ (18) are $k_1 = 0.6$, $k_2 = 0.2$, $k_3 = 0.1$, $k_4 = 0.1$ and $k_h = 0.1$ (17). The minimum set of obtained waypoints has $n_w = 19$ elements. The minimum set of waypoint obtained by OMWS-ET is smaller than the method OMWS-GM which does not consider the orientation neither the vehicle's model. To avoid a large growing of the tree branch of OMWS-ET, a position and orientation comparison between nodes can be added at line 13 of the Algorithm 3. If two nodes from different branches have the same position and orientation then the node with lowest total cost function value (cf. (11)) is stored and the other node is removed.

Table 1 shows different performance criteria to compare the set of waypoints where: n_w is the number of waypoints, $length$ is the sum of distance between two successive waypoints, d_{border} is the sum of minimum distance to the road boundaries. Therefore, the method based on OMWS-ET is more safe, accurate and efficient than the one based on OMWS-GM, mainly when the criterion to optimize is related to the vehicle's model (velocity and steering angle).

*4.1.2 OMWS-ET versus RRT**

To highlight the advantages and the flexibility of the proposed OMWS-ET, a comparison with the popular

Table 1 Comparison between the OMWS-GM and ET

	n_w	$length$[m]	d_{border}[m]
OMWS-GM	27	77.22	52.1268
OMWS-ET	19	77.50	56.7217

RRT* algorithm [15] is presented in this subsection. The RRT* is based on the RRT (Rapidly-exploring Random Tree) already described in Section 1.2 with an addition of the rewiring function which allows to reconnect the nodes to ensure that the edges have the path with minimum total cost. RRT* provides thus an optimal solution with minimal computational and memory requirements [15]. Moreover, RRT* is a sampling-based algorithm and the optimal solution depends on the number of iterations of the RRT* algorithm, i.e., more is the number of iterations (more samples in the C-space$_{free}$) closer is the obtained solution to the effective optimal global solution. Therefore, to compare the solutions obtained by the OMWS-ET with those obtained by the RRT* some little modifications in Algorithm 3 were made. The line 6 of Algorithm 3 was changed by a for loop from 0 to the maximum iteration number and the selection of the final pose at each iteration is obtained by the sampling in C-space$_{free}$ (q_{random}) as the RRT* Algorithm [15]. It is to be noted that q_{random} corresponds to a random sample (position) from a uniform distribution in the C-space$_{free}$.

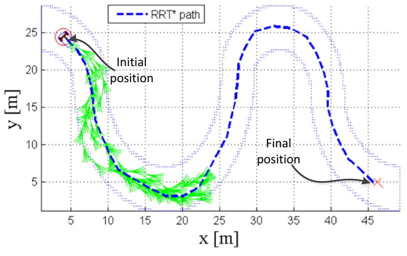
To compare the two algorithms (OMWS-ET and RRT*), the safest obtained solution (wich maximizes the distance to the border) is used as criterion. Therefore, the parameters of the cost function of OMWS-ET (18) are fixed to: $k_1 = 1.0$, $k_2 = k_3 = k_4 = 0.0$ and $k_h = 0.1$. In addition, the other parameters are fixed as: the branch number $n_t = 5$; the edge distance $\xi = 2.5\ m$ and $\Delta\alpha = 15°$. The RRT* algorithm described in [15] was also modified to obtain a cost function according to the safety $\bar{w}_i$ (distance to the border) instead of an Euclidean distance between nodes. The kinematic model (5) with constant linear velocity and steering angles ($v = 1.0\ m/s$ and $\gamma = -15, -7.5, 0, 7.5, 15°$) respectively, during $t_{exp} = 2.5\ s$ was used to produce the new nodes of the RRT* (cf. Section 1.2). The maximum number of iteration for both algorithm is fixed to $n_I = 5000$.

Table 2 Comparison between Voronoï, RRT* and OMWS-ET

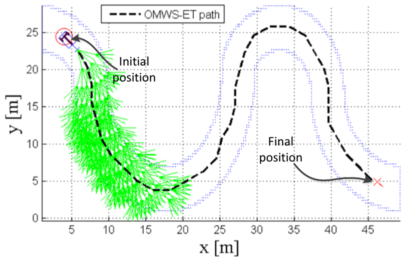
	$length$[m]	d_{border}[m]
Voronoï	86.00	69.2931
RRT*	83.42	62.1736
OMWS-ET	82.50	65.5926

Figure 15 shows the obtained path solutions according to RRT*, OMWS-ET and also to Voronoï [20] algorithms. The Voronoï obtained path (cf. Fig. 15c) is given only because it is the best reference w.r.t. the adopted comparison criterion (safety criterion). Indeed, Voronoï path permits always to obtain the safest possible path [20] It can be noted that the two obtained path using RRT* and OMWS-ET are generally enough close and far from the way border (cf. Fig. 15a and b). Important differences are nevertheless observed in the obtained final results (cf. Fig. 15c). In fact, the obtained set of waypoint using RRT* are closer to the border which is due to the fact that RRT* expands its branches while adopting constant commands (v, γ, t_{exp}). These constant commands generate the next nodes with only a single possible orientation (for each node). Contrary to that, in the proposed OMWS-ET, each new obtained node q_j has different possible orientations and velocities, thus, for the same position, much more possible vehicle's states (different orientations and velocity setpoints) are taking into account in the optimization process.

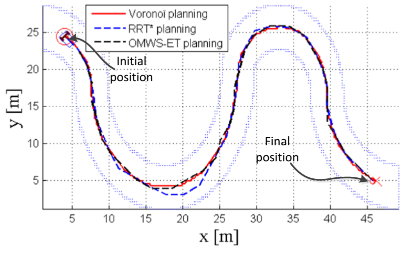
Table 2 shows, as in the last subsection, different performance criteria to compare the obtained path. It is shown that the obtained path based on OMWS-ET



(a) RRT* with a few branches



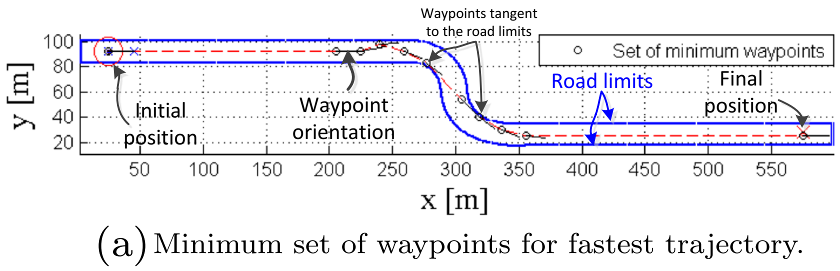
(b) OMWS-ET with a few branches



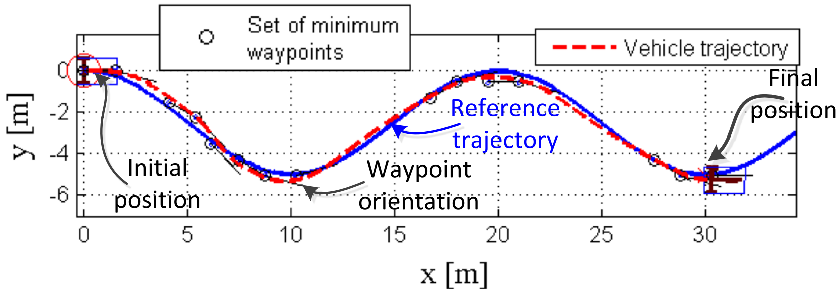
(c) Comparison with Voronoï

Fig. 15 Three obtained path according to Voronoï, RRT* and OMWS-ET

Fig. 16 Different scenario for OMWS-ET



(a) Minimum set of waypoints for fastest trajectory.



(b) Minimum set of waypoints in a reference path.

is closer than the RRT* to the optimal obtained solution using Voronoï methodology. It validates that the proposed OMWS-ET is more efficient than the RRT*, in the sens that it explores much more possibilities in the vehicle/environment/task state space.

It is important to mention also, that the proposed OMWS-ET methodology is related to the adopted navigation strategy (cf. Section 2), which uses set-points based on suitable static/dynamic waypoints instead of trajectory tracking methods. OMWS-ET method takes into account the vehicle's kinematics constraints and uncertainites as well as the used control law (cf. Section 3.2). RRT* method is more suitable for navigation strategies based on trajectory following [15].

4.1.3 Specific Scenario Cases

We show in what follows other minimum set of obtained waypoints for different scenarios using the method based on expanding tree (Algorithm 3). Figure 16a shows the set of waypoints while considering the edge distance $\xi = 10\ m$ with an objective to obtain the fastest trajectory from the initial to the final positions, while not colliding with the road limits. The constant values are $k_1 = 0.1$, $k_2 = 0.7$, $k_3 = 0.1$, $k_4 = 0.1$ and $k_h = 0.1$. The minimum set of waypoints allows the vehicle to generate a minimum time trajectory as in [26]. This trajectory has a segment close to the route boundaries (tangent to the borders) which allows to navigate applying the maximum velocity.

Figure 16b shows the use of the proposed OMWS-ET for the specific case where a reference path already exists for the navigation of the vehicle. In this case, the set of waypoints will be chosen as close as possible to the considered path (depends on the chosen values of ξ and $\Delta\alpha$ in Algorithm 3). The set of waypoints obtained using OMWS-ET allows thus more flexible and safe navigation of the vehicle between the waypoints (cf. the criterion to optimize in Eq. 18). The edge distance ξ is set to $1\ m$. The minimum set of waypoints are obtained while considering the term $\bar{w}_j$ (18) as the normalized minimum distance of the node q_j to the reference path. The constant values are set to $k_1 = 0.6$, $k_2 = 0.2$, $k_3 = 0.1$, $k_4 = 0.1$

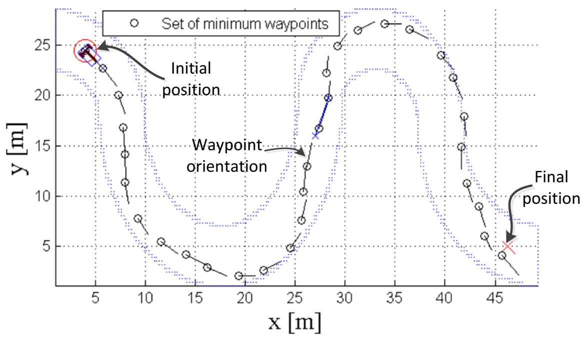


Fig. 17 Set of waypoints using probabilistic expanding tree

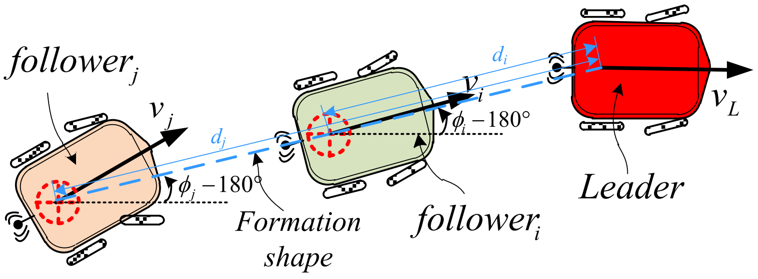


Fig. 18 Multi-robot formation (straight line shape)

and $k_h = 0.1$. The values of ξ and $\Delta\alpha$ can produce some waypoints outside the reference trajectory, e.g., if we decrease the values of ξ and $\Delta\alpha$ and increase the number of branches n_t then the waypoints will be on the reference trajectory. In [33], the waypoints are selected while considering only the points in the reference trajectory. It consists on analyzing the orientation variation of each points on the trajectory. In our case, the waypoints are selected in the environment to be close to the reference trajectory which allows to obtain less number of waypoints than the method used in [33].

4.1.4 Deterministic versus Probabilistic

This simulation shows the comparison between a deterministic and probabilistic expanding tree (i.e., where the values of ξ and α are probabilistically taken from an interval, instead of, to be fixed by the designer). Figure 17 shows the minimum set of waypoints obtained using probabilistic expanding tree, where $\xi \in [0,\ 2.5]$ and $\alpha \in [-30°, 30°]$. The constant values are $k_1 = 0.6$, $k_2 = 0.2$, $k_3 = 0.1$, $k_4 = 0.1$ and $k_h = 0.1$. The processing time of the method with probabilistic expanding tree is less than the method with deterministic expanding tree. Nevertheless, the set of waypoints are not the optimal solution. The advantages of probabilistic selection of ξ and α is to reduce the convergence time and to obtain an online implementation [18, 32]. In future works, the choice of the variation of ξ and α will be oriented to improve the efficiency of the algorithm.

4.1.5 Extension to Multi-robot Formation

Our method based on expanding tree (Algorithm 3) was extended to multi-robot formation where the

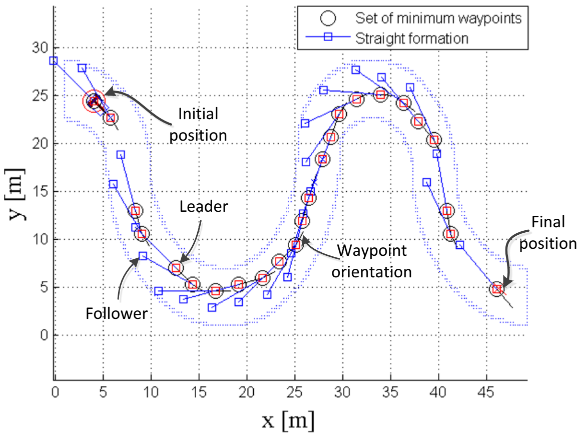


Fig. 19 Minimum set of waypoints for multi-robot formation obtained by Algorithm 3 based on expanding tree

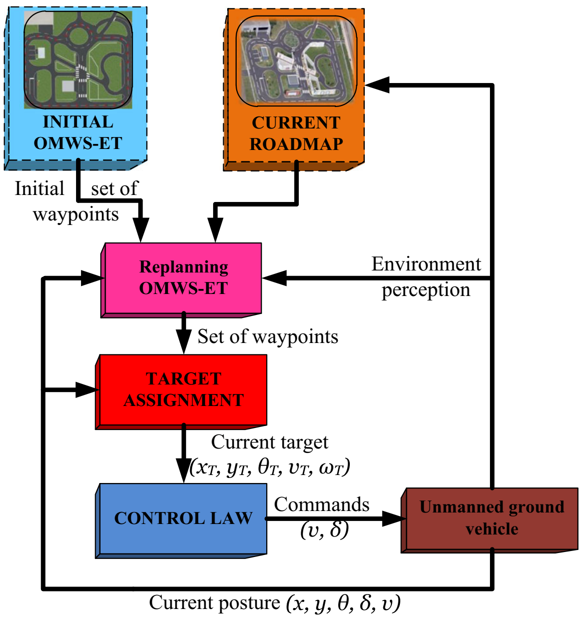
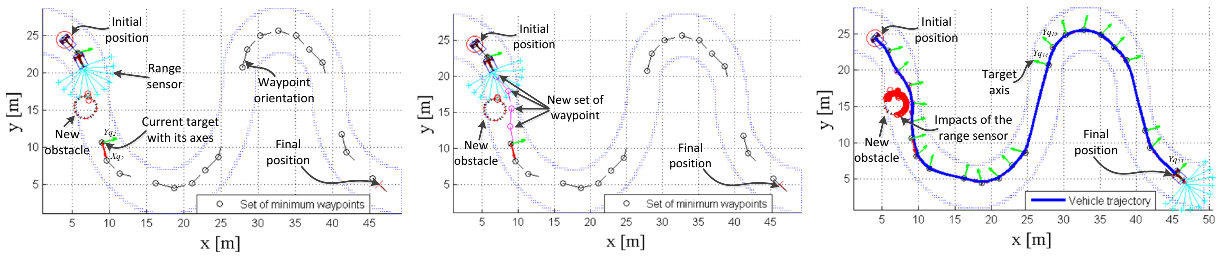


Fig. 20 Schema of the local replanning



(a) Unexpected obstacle is detected. (b) Local replanning. (c) Safe vehicle trajectory.

Fig. 21 Local replanning for unexpected obstacle

formation is defined only according to the leader configuration [9] (cf. Fig. 18). As mentioned before, the OMWS-ET algorithm takes into account the vehicle model. To cope with this multi-robot task, it is sufficient to adapt the term $\Delta\bar{e}_{l_i j}$ (24) in order to consider all trajectories of the group of UGVs. Figure 19 shows the minimum set of waypoints for a line formation ($d_i = 6\,m$ and $\phi_i = 180°$) with two vehicles. The constant values are the same as the last simulation. The set of waypoints for the leader UGV are close to the curve road boundaries because the formation needs enough space to turn while keeping the rigid formation shape. The follower (blue square) is always inside of the road boundaries.

4.1.6 Local Replanning for Unexpected Obstacles

The proposed method OMWS-ET is adapted to local replanning when an unexpected static obstacle is detected in the environment. Figure 20 shows the used architecture to activate the replanning of the vehicle's movements based on an initial set of waypoints already obtained using OMWS-ET. The vehicle starts the navigation through the successive waypoints (cf. Section 2.1) from the initial set of waypoints. They were already computed using the OMWS-ET in the known environment (cf. Section 3.2). The vehicle uses a range sensor to detect any unforeseen obstacle (cf. Fig. 21a). A local replanning is activated when any new obstacle is detected. This replanning takes into account the current environment state, the current vehicle pose and the current waypoint to obtain a new local set of waypoints (cf. Fig. 21b). If the current waypoint is unreachable (due to the presence of the obstacle) then the final position is replaced by the next waypoint in the list and so on. If no solution is found then the vehicle will stop in its current pose. Figure 21b shows an example of the local replanning using the set of waypoints given in Section 4.1.1 (as initial set of waypoints)

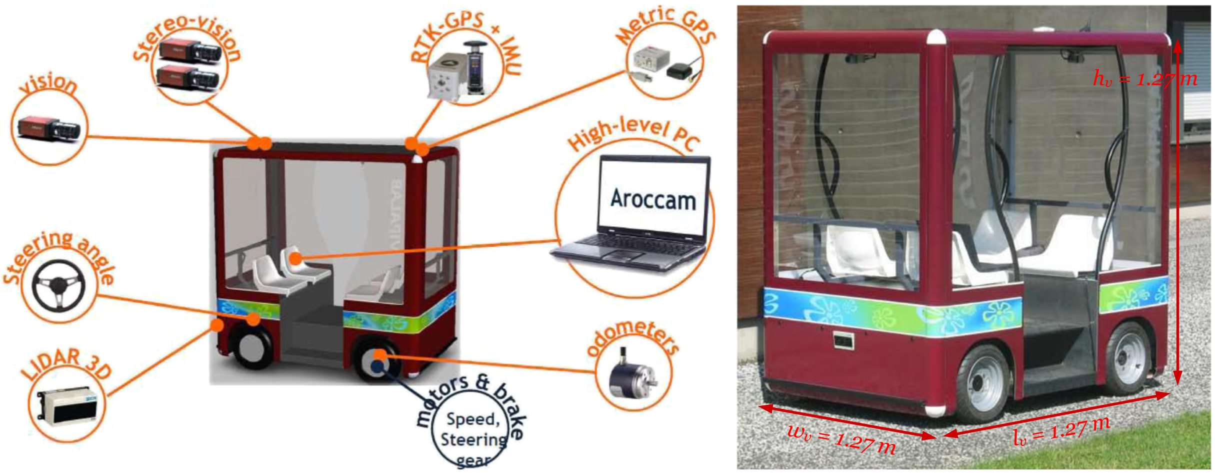


Fig. 22 VIPALAB electric urban vehicle

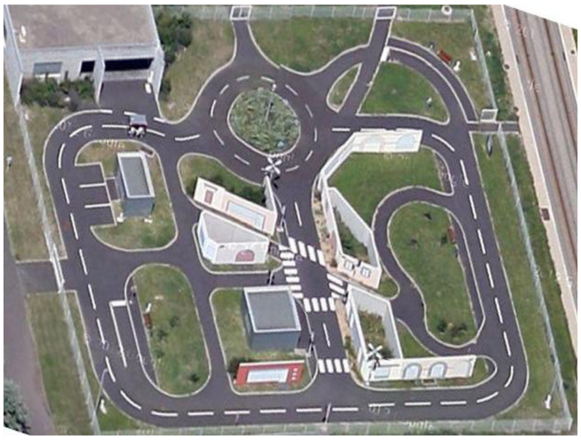


Fig. 23 PAVIN experimental platform (Clermont-Ferrand, France)

(cf. Fig. 14b). Finally, the vehicle moves through the new set of waypoints while guaranteeing a safe navigation (cf. Fig. 21c).

4.2 Experimental Results

The navigation strategy and the proposed method based on expanding tree was also experimented with the actual VIPALAB vehicle (cf. Fig. 22) in an experimental environment named PAVIN (*Plate-forme d'Auvergne pour Véhicules INtelligents*) (cf. Fig. 23). These experiments can be found online.[1] This vehicle carry different embedded proprioceptive and exteroceptive sensors such as odometers, gyrometer, steering angle sensor and an RTK-GPS. Each vehicle uses a combination of RTK-GPS and gyrometer to estimate its current position and orientation at a sample time of $T_s = 0.01\ s$.

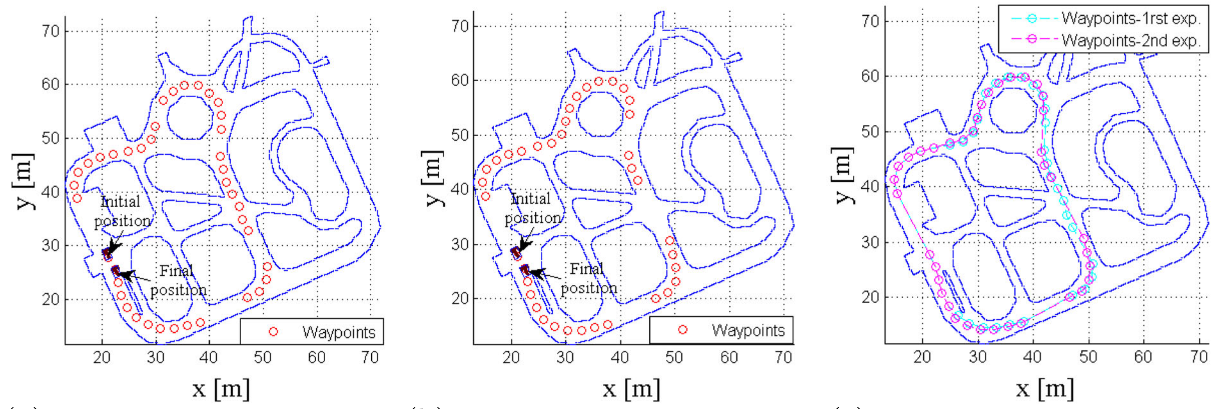
A metric map of the environment PAVIN [12] is used by the proposed method (Algorithm 3). This map allows to implement the navigation through successive waypoints in a real vehicle (cf. Section 2.1). The proposed method based on expanding tree computes the set of geo-referenced waypoints with optimal configuration. Certain areas are restricted to guide the Algorithm 3 through PAVIN platform which has intersections and roundabout (cf. Fig. 24). In our case, these restricted areas were selected by the user, nevertheless the selection can be made by considering the topological map of the environment. We experiment the proposed OMWS-ET to make a comparison between two cases: the first, corresponds to give more priority for the safety criteria in Eq. 18 and the second gives more priority for the minimum angle steering rate. The analysis of the obtained solutions will be given in what follows. Moreover, the actual vehicle's trajectories are compared for these different set of waypoints.

Figures 24 and 25 show respectively the minimum obtained set of waypoints and the corresponding vehicle's trajectories (in simulation and actual experiment). Figure 24a shows the set of waypoints of the first experiment where the constant values of the cost function (18) are $k_1 = 0.6$, $k_2 = 0.2$, $k_3 = 0.1$, $k_4 = 0.1$ and $k_h = 0.4$. The safety (k_1) has the highest priority in this experiment. Therefore, these waypoints guide the vehicle to be close to the middle of the route (cf. Fig. 25a). Figure 24b shows the set of waypoints of the second experiment where the constant values are $k_1 = 0.3$, $k_2 = 0.2$, $k_3 = 0.4$, $k_4 = 0.1$ and $k_h = 0.4$. The minimal steering angle rate k_3 has the highest priority in this experiment. The obtained result shows that the obtained waypoints are localized very close to the border of the road (cf. Fig. 25b). Figure 25a and b show the simulated and the actual vehicle trajectories. It can be observed that they are very close (maximal error between them is less than 0.15 m). We can conclude thus that the proposed optimal multi-criteria waypoint selection based on Expanding Tree (OMWS-ET, performed off-line (cf. Section 3.2)) permits to cope accurately with actual environment and experiments.

Figures 24c and 25c show the comparison between the set of waypoints and the real trajectories of both experiments. The velocities and steering angle of the vehicle while tracking each waypoint are shown in Fig. 26. This figure shows the values with noise due to the encoder inaccuracies.

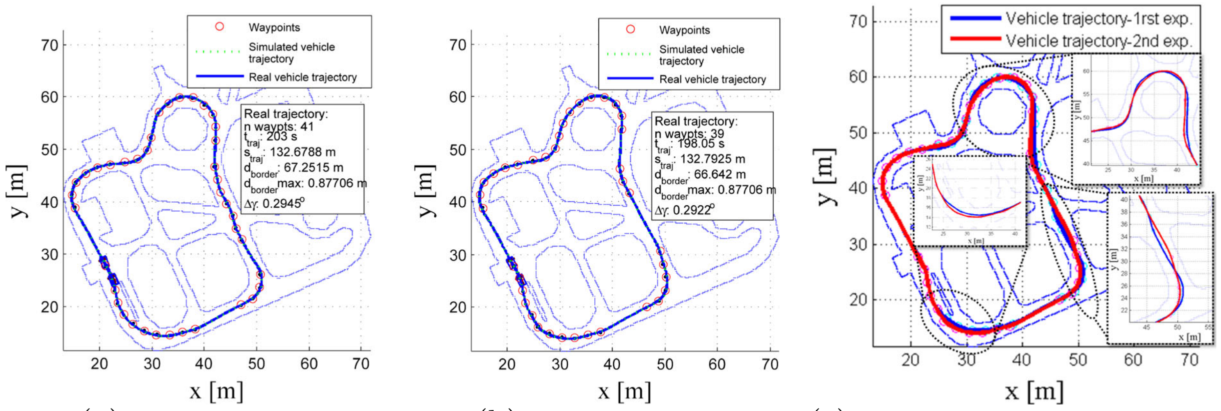
Table 3 shows different performance criteria to compare the set of waypoints where: n_w is the number of waypoints, T is the navigation time, l_{UGV} is the traveled distance, d_{border} is the sum of minimum distance to the road boundaries and $\Delta\gamma$ is the root mean square (rms) of the steering angle rate. We note that the first experiment has n_w greater than the second experiment. It is due to the fact that the first experiment has the safety as a priority. The proposed Algorithm 3 selects thus more waypoints to allows the vehicle to navigate as farther as possible

[1] http://maccs.univ-bpclermont.fr/uploads/Profiles/VilcaJM/OMWS.mp4



(a) First experiment: Safe planning. (b) Second experiment: Steering angle minimization. (c) Comparison between experiments.

Fig. 24 Different set of obtained waypoints



(a) First experiment. (b) Second experiment. (c) Comparison between actual experiments.

Fig. 25 Actual vehicle's trajectories for different obtained set of waypoints

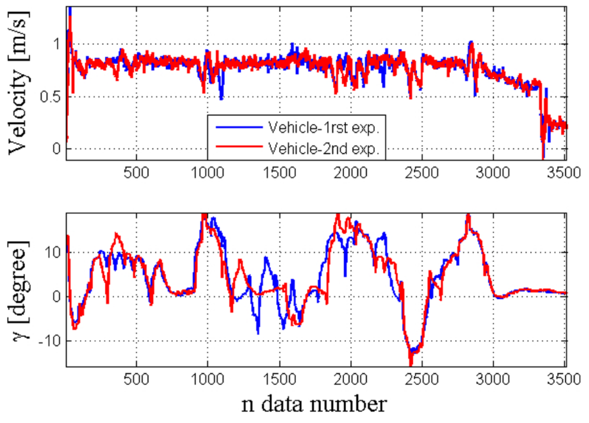


Fig. 26 Vehicle velocities and steering angles progress for each set of obtained waypoints

Table 3 Comparison among the set of waypoints

		n_w	$T[s]$	l_{UGV}[m]	d_{border}[m]	$\Delta\gamma[°]$
1rst	Sim.	41	200	132.81	67.35	0.3123
exp.	Real	41	203	132.68	67.25	0.2945
2nd	Sim.	39	199	133.00	66.54	0.3089
exp.	Real	39	198	132.79	66.64	0.2922

from the road borders. It can be noticed by d_{border} where its value is bigger in the first experiment than the second. Furthermore, the values of $\Delta\gamma$ is less in the second experiment because the highest priority was for the steering angle rate. Therefore, the vehicle can navigate with higher velocity along the trajectory and the navigation time is smaller than the first experiment.

5 Conclusion

This paper has presented two planning methods to obtain the optimal waypoints configuration (Optimal Multi-criteria Waypoints Selection based on Expanding Tree (OMWS-ET) and Grid Map (OMWS-GM)) which guarantees safe, smooth and feasible vehicle navigation in a structured environment. The flexible navigation strategy throughout optimal and discrete selected waypoints was also presented. It allows to avoid any trajectory planning which could be time consuming. The proposed OMWS-GM is based on the A* algorithm with an additional term to consider the orientation change between successive cells. OMWS-ET uses a multi-criteria function which takes into account the vehicle model and uncertainties to obtain the optimal set of waypoints configurations (position, orientation and velocity). Moreover, it has been shown that the proposed OMWS-ET is much more accurate and flexible than OMWS-GM. A multitude of simulations and experimental results demonstrate the efficiency and reliability of the proposed OMWS-ET in different cases (trajectory specification, deterministic versus probabilistic, comparison with RRT*, multi-robot task, local replanning according to the multi-criteria optimization).

In future works, an extension using the dynamic model of the vehicle and $3D$ position will be developed (notably for unmanned aerial vehicle). The enhancement of the proposed methods for hard real time application will be developed. Genetic algorithm will be notably investigated. In addition, we will extend the proposed strategies for robust navigation in formation of a group of robots.

Acknowledgments Supported by the French National Agency (ANR) through the SafePlatoon project and LABEX IMobS3 (ANR-7107-LABX-716701).

Appendix

This section described briefly the stability analysis based on Lyapunov method used to demonstrate the convergence of the vehicle to the target posture, i.e., for a finite time, the error system (e_x, e_y, e_θ) converges to zero [16]. Let us first define the Lyapunov function V by Eq. 26. It is a function of three parameters which depend on: the distance d between the target and vehicle positions, the distance d_l from the vehicle to the target line (line that pass through the target position with orientation equal to the target orientation), this term is related to the Line of Sight and Flight of the target, and the orientation error e_θ between the vehicle and the target (cf. Fig. 5). It is represented by:

$$\begin{aligned} V &= \frac{1}{2}K_d d^2 + \frac{1}{2}K_l d_l^2 + K_o[1-\cos(e_\theta)] \\ &= \frac{1}{2}K_d d^2 + \frac{1}{2}K_l d^2 \sin^2(e_{RT}) \\ &\quad + K_o[1-\cos(e_\theta)] \end{aligned} \tag{26}$$

where the initial values of e_{RT} and e_θ satisfy:

$$e_{RT} \in \left]-\pi/2, \pi/2\right[\ \ and \ \ e_\theta \in \left]-\pi/2, \pi/2\right[\tag{27}$$

These conditions (27) guarantee that the target is ahead to the vehicle w.r.t. its orientation. Moreover, Eq. 27 has open interval that allows to avoid local minimum. Therefore, V is a positive-definite function [16].

The Lyapunov function (26) can be written according to e_x, e_y as follows:

$$V = \frac{1}{2}\left(e_x^2 + e_y^2\right)[K_d + K_l \sin^2(e_{RT})] + K_o[1-\cos(e_\theta)] \tag{28}$$

To guarantee the system stability, $\dot{V}$ has to be negative-definite [16]. By taking the derivative of Eqs. 2, 4 and 28 and using Eqs. 7 and 8, $\dot{V}$ can be written:

$$\begin{aligned} \dot{V} &= (e_x\dot{e}_x + e_y\dot{e}_y)\left[K_d + K_l\sin^2(e_{RT})\right] \\ &\quad + K_l d^2 \sin(e_{RT})\cos(e_{RT})\dot{e}_{RT} + K_o\sin(e_\theta)\dot{e}_\theta \\ &= [-e_x v_b + v_T e_y \sin(e_\theta)][K_d + K_l\sin^2(e_{RT})] \\ &\quad + K_l \sin(e_{RT})\cos(e_{RT})\left[\frac{d^2 v_T}{r_{c_T}} - v_T e_x \sin(e_\theta) \right. \\ &\quad \left. - e_y v_b\right] \end{aligned} \tag{29}$$

$$+K_o\sin(e_\theta)\left(\frac{v_T}{r_{c_T}} - v_T\cos(e_\theta)c_c - v_b c_c\right) \tag{30}$$

Using Eq. 6 in the first two terms of Eq. 30 and factorizing the common terms, it holds that:

$$\begin{aligned}\dot{V} &= v_T \sin(e_\theta)[K_d e_y - K_l d \sin(e_{RT})\cos(e_\theta)] \\ &\quad + \frac{v_T}{r_{c_T}}[d^2 K_l \sin(e_{RT})\cos(e_{RT}) + K_o \sin(e_\theta)] \\ &\quad - v_b\big[K_d e_x + K_l d \sin(e_{RT})\sin(e_\theta) \\ &\quad + K_o \sin(e_\theta) c_c\big] \quad (31) \\ &\quad - v_T K_o \sin(e_\theta)\cos(e_\theta) c_c \quad (32)\end{aligned}$$

Finally, using Eqs. 9 and 10 in Eq. 32, we obtain:

$$\begin{aligned}\dot{V} &= -K_x\big[K_d e_x + K_l d \sin(e_{RT})\sin(e_\theta) \\ &\quad + K_o \sin(e_\theta) c_c\big]^2 - v_T K_o K_\theta \sin^2(e_\theta) \quad (33) \\ &\quad - v_T K_o K_{RT} \sin^2(e_{RT}) \le 0 \quad (34)\end{aligned}$$

Equation 34 shows that the system is stable while the initial conditions (27) are satisfied. To ensure the asymptotic stability of the error system, $\dot{V}$ has to be a negative-definite function. Let us exhibit the case where $\dot{V} = 0$ with $v_T > 0$ and $v_T = 0$. Firstly, when $v_T > 0$ and using the initial assumption $\mathbf{K} > 0$, it is straightforward to show that e_x, e_θ, e_{RT} are equal to zero to satisfy Eq. 34, then according to Eqs. 4, 5 and 27 d is equal to zero ($e_y = 0$). Hence, $\dot{V}$ is equal to zero when $v_T > 0$, only if $(e_x, e_y, e_\theta) = (0, 0, 0)$.

Secondly, let us consider the case where $v_T = 0$. The initial assumption is identical. Hence, the second and third terms of Eq. 34 are equal to zero when $v_T = 0$. Additionally, when $v_T = 0$, we consider that $r_{c_T} \to \infty$, consequently the first term of $\dot{V}$ is equal to zero when:

$$K_d e_x + K_l d \sin(e_{RT})\sin(e_\theta) + K_o \sin(e_\theta) c_c = 0 \quad (35)$$

Replacing Eq. 10 with $r_{c_T} \to \infty$ in Eq. 35, the following expression is obtained:

$$\begin{aligned}0 &= K_d e_x + K_l d \sin(e_{RT})\sin(e_\theta) \\ &\quad + \tan(e_\theta)[K_d e_y - K_l d \sin(e_{RT})\cos(e_\theta)] \\ &\quad + K_o \sin(e_\theta)\left[K_\theta \tan(e_\theta) + \frac{K_{RT}\sin^2(e_{RT})}{\sin(e_\theta)\cos(e_\theta)}\right] \\ &= K_d[e_x + e_y \tan(e_\theta)] + K_o K_\theta \frac{\sin^2(e_\theta)}{\cos(e_\theta)} \\ &\quad + K_o K_{RT} \frac{\sin^2(e_{RT})}{\cos(e_\theta)} \quad (36)\end{aligned}$$

Using Eq. 6 in Eq. 36, we obtain:

$$K_d d \frac{\cos(e_{RT})}{\cos(e_\theta)} + K_o K_\theta \frac{\sin^2(e_\theta)}{\cos(e_\theta)} + K_o K_{RT} \frac{\sin^2(e_{RT})}{\cos(e_\theta)} = 0 \quad (37)$$

Equation 37 exhibits quadratic terms. Consequently, considering the initial conditions (27), $\cos(e_{RT})$ and $cos(e_\theta)$ are greater than zero. Therefore, all the terms of Eq. 37 are positive and they must be equal to zero, i.e., $d = e_\theta = e_{RT} = 0$, and if $d = 0$ then $e_x,\ e_y = 0$. Hence, from Eq. 37, $\dot{V}$ is equal to zero when $v_T = 0$ and $r_{c_T} \to \infty$, only if $(e_x, e_y, e_\theta) = (0, 0, 0)$.

Conclusively, if $v_T > 0$ and $v_T = 0$, V is always strictly positive and $\dot{V}$ is always strictly negative while $(e_x, e_y, e_\theta) \neq (0, 0, 0)$. Therefore, the errors system is asymptotically stable while the initial vehicle conditions (27) are satisfied.

References

1. Abbadi, A., Matousek, R., Petr Minar, P.S.: RRTs review and options. In: Proceedings of the International Conference on Energy, Environment, Economics, Devices, Systems, Communications, Computers (2011)
2. Adouane, L., Benzerrouk, A., Martinet, P.: Mobile robot navigation in cluttered environment using reactive elliptic trajectories. In: Proceedings of the 18th IFAC World Congress (2011)
3. Aicardi, M., Casalino, G., Bicchi, A., Balestrino, A.: Closed loop steering of unicycle like vehicles via lyapunov techniques. J. Math. Mech. IEEE **2**(1), 27–35 (1995)
4. Bellman, R.: A markovian decision process. J. Math. Mech. **6**, 679–684 (1957)
5. Bertsekas, D.P.: Dynamic Programming and Optimal Control, vol. I. Athena Scientific (1995)
6. Bonfè, M., Secchi, C., Scioni, E.: Online trajectory generation for mobile robots with kinodynamic constraints and embedded control systems. In: Proceedings of the 10th International IFAC Symposium on Robot Control. Croatia (2012)
7. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementation. MIT Press (2005)
8. Connors, J., Elkaim, G.H.: Manipulating b-spline based paths for obstacle avoidance in autonomous ground vehicles. In: Proceedings of the ION National Technical Meeting, ION NTM 2007 San Diego (2007)
9. Consolini, L., Morbidi, F., Prattichizzo, D., Tosques, M.: Leader-follower formation control of nonholonomic mobile robots with input constraints. Automatica **44**(5), 1343–1349 (2008)
10. Gu, T., Dolan, J.M.: On-road motion planning for autonomous vehicles. In: Su, C.Y., Rakheja, S., Liu, H. (eds.) Intelligent Robotics and Applications, vol. 7508. Springer Berlin Heidelberg (2012)
11. Horst, J., Barbera, A.: Trajectory generation for an on-road autonomous vehicle. Proceedings of the SPIE, Unmanned Systems Technology VIII (2006)

12. The Institut Pascal Data Sets. http://ipds.univ-bpclermont.fr (2013)
13. Jazar, R.N.: Vehicle Dynamics: Theory and Application, Chapter 7. Springer-Verlag (2014)
14. Kallem, V., Komoroski, A., Kumar, V.: Sequential composition for navigating a nonholonomic cart in the presence of obstacles. IEEE Trans. Robot. **27**(6), 1152–1159 (2011)
15. Karaman, S., Frazzoli, E.: Sampling-based Algorithms for Optimal Motion Planning. Int. J. Robot. Res. **30**(7), 846–894 (2011)
16. Khalil, H.K.: Nonlinear Systems. Prentice Hall (2002). (1986)
17. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Res. **5**, 90–99 (1986)
18. Kuwata, Y., Fiore, G.A., Teo, J., Frazzoli, E., How, J.P.: Motion planning for urban driving using rrt. In: International Conference on Intelligent Robots and Systems, pp. 1681–1686 (2008)
19. Labakhua, L., Nunes, U., Rodrigues, R., Leite, F.: Smooth trajectory planning for fully automated passengers vehicles: Spline and clothoid based methods and its simulation. In: Cetto, J., Ferrier, J.L., Costa dias Pereira, J.M., Filipe, J. (eds.) Informatics in Control Automation and Robotics, Lecture Notes Electrical Engineering, vol. 15, pp. 169–182. Springer Berlin Heidelberg (2008)
20. Latombe, J.C.: Robot Motion Planning. Kluwer Academic Publishers, Boston (1991)
21. LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)
22. Lee, J.W., Litkouhi, B.: A unified framework of the automated lane centering/changing control for motion smoothness adaptation. In: Proceedings of the 15th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 282–287 (2012)
23. Luca, A.D., Oriolo, G., Samson, C.: Feedback control of a nonholonomic car-like robot. In: Laumond, J.P. (ed.) Proceedings of the Robot Motion Planning and Control, pp. 171–253. Springer-Verlag, Berlin (1998)
24. Maalouf, E., Saad, M., Saliah, H.: A higher level path tracking controller for a four-wheel differentially steered mobile robot. Robot. Auton. Syst. **54**, 23–33 (2006)
25. Martins, M.M., Santos, C.P., Frizera-Neto, A., Ceres, R.: Assistive mobility devices focusing on smart walkers: Classification and review. Robot. Auton. Syst. **60**(4), 548–562 (2012)
26. Rucco, A., Notarstefano, G., Hauser, J.: Computing minimum lap-time trajectories for a single-track car with load transfer. In: Decision and Control (CDC), 2012 IEEE 51st Annual Conference on, pp. 6321–6326 (2012)
27. Sezen, B.: Modeling automated guided vehicle systems in material handling. Otomatiklestirilmi Rehberli Arac Sistemlerinin Transport Tekniginde Modellemesi. Dou Universitesi Dergisi **4**(2), 207–216 (2011)
28. Sharma, S., Taylor, M.E.: Autonomous waypoint selection for navigation and path planning: A navigation framework for multiple planning algorithms. Tech. Rep. (2012)
29. Siciliano, B., Khatib, O.: (eds.): Springer Handbook of Robotics, Part E-34. Springer (2008)
30. Stoeter, S.A., Rybski, P.E., Stubbs, K.N., McMillen, C.P., Gini, M., Hougen, D.F., Papanikolopoulos, N.: A robot team for surveillance tasks: Design and architecture. Robot. Auton. Syst. **40**(2-3), 173–183 (2002)
31. Szczerba, R., Galkowski, P., Glicktein, I., Ternullo, N.: Robust algorithm for real-time route planning. IEEE Trans. Aerosp. Electron. Syst. **36**(3), 869–878 (2000)
32. Vaz, D.A., Inoue, R.S., Grassi Jr. V.: Kinodynamic motion planning of a skid-steering mobile robot using rrts. In: Proceedings of the 2010 Latin American Robotics Symposium and Intelligent Robotics Meeting, LARS '10, pp. 73–78. IEEE Computer Society (2010)
33. Vilca, J., Adouane, L., Mezouar, Y., Lébraly, P.: An overall control strategy based on target reaching for the navigation of an urban electric vehicle. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13). Tokyo (2013)
34. Ziegler, J., Werling, M., Schroeder, J.: Navigating car-like robots in unstructured environment using an obstacle sensitive cost function. In: Proceedings of the IEEE Intelligent Vehicle Sympsium (IV), pp. 787–791. Netherlands (2008)

José Vilca is currently a Ph.D. student in the Institut Pascal, UMR 6602, CNRS/Université Blaise Pascal, France. He received his M.Sc. in Dynamic System from University of São Paulo, Brazil, in 2011 and his B.Eng. degree in Electronic Engineering from National University of Engineering, Peru, in 2006. He belongs to the Image, Perception Systems, Robotics Group at Institut Pascal. His research interests include cooperative systems, hybrid control systems, multi-robot coordination, robotics, mobile-robot autonomous navigation and nonlinear control.

Lounis Adouane received his Master of Sciences in 2001 from IRCCyN - ECN Nantes (France), where he worked on the control of legged mobile robotics. In 2005 he obtained the Ph.D. in Automatic Control from LAB - UFC Besançon. During his PhD Lounis Adouane has deeply investigated the field of multi-robot systems, especially those relaying to reactive control architectures. After that, he joined in 2006 LAI - INSA Lyon and he studied the hybrid architecture of control applied to cooperative mobile arms robots. Since 2006, he is an Associate Professor at Institut Pascal - Polytech Clermont-Ferrand. His research interests include: Mobile robotics control, Cooperative robotics, Artificial intelligence, Behavioral/Hybrid control architectures and Multi-robot simulation.

Youcef Mezouar received the Ph.D. degree in computer science from the Université de Rennes 1, Rennes, France, in 2001 and the "Habilitation à Diriger les Recherches" degree from Université Blaise Pascal, Clermont-Ferrand, France, in 2009. He is currently a Professor at Institut Français de Mécanique Avancée and a member of Institut Pascal, UMR 6602, CNRS/Université Blaise Pascal, France. He is a Coleader of the Image, Perception Systems, Robotics Group, Institut Pascal, and he leads the Modeling, Autonomy and Control in Complex Systems Team at Institut Pascal. His research interests include automatics, robotics, and computer vision, particularly visual servoing and mobile-robot navigation.