Noname manuscript No.
(will be inserted by the editor)

A Real-Time Path-Planning Algorithm based on Receding Horizon Techniques

M.Murillo · G.Sánchez · L.Genzelis · L.Giovanini

Received: date / Accepted: date

Abstract In this article we present a real-time path-planning algorithm that can be used to generate optimal and feasible paths for any kind of unmanned vehicle (UV). The proposed algorithm is based on the use of a simplified particle vehicle (PV) model, which includes the basic dynamics and constraints of the UV, and an iterated non-linear model predictive control (NMPC) technique that computes the optimal velocity vector (magnitude and orientation angles) that allows the PV to move towards desired targets. The computed paths are guaranteed to be feasible for any UV because: i) the PV is configured with similar characteristics (dynamics and physical constraints) as the UV, and ii) the feasibility of the optimization problem is guaranteed by the use of the iterated NMPC algorithm. As demonstration of the

M.Murillo · G.Sánchez · L.Genzelis · L.Giovanini
Instituto de Investigación en Señales, Sistemas e Inteligencia Computacional, sinc(i), UNL, CONICET, Ciudad Universitaria UNL, 4to piso FICH, (S3000) Santa Fe, Argentina.
Tel: +54(342)4575233/34 ext 118. Fax: +54(342)4575224.
E-mail: mmurillo@sinc.unl.edu.ar · gsanchez@sinc.unl.edu.ar · lgenzelis@sinc.unl.edu.ar · lgiovanini@sinc.unl.edu.ar

capabilities of the proposed path-planning algorithm, we explore several simulation examples in different scenarios. We consider the existence of static and dynamic obstacles and a follower condition.

Keywords feasible optimal path · model predictive control · real-time path-planning · replanning

1 Introduction

One of the areas that has grown surprisingly fast in the last decade is the one involving autonomous Unmanned Vehicles (UVs), both aerial (*Unmanned Aerial Vehicles* - UAVs) and terrestrial (*Unmanned Ground Vehicles* - UGVs). Their reduced size and geometry allow them to carry out dangerous missions at lower costs than their manned counterparts without compromising human lives. They are mostly used in missions such as search and rescue, power line inspections, precision agriculture, imagery and data collection, security applications, mine detection and neutralization, operations in hazardous environments, among others [1–6]. In general, most of such missions require that the UVs move in uncertain scenarios avoiding different types of obstacles. To do so, they must have the ability to *autonomously* determine and track a feasible collision-free path.

The path-planning problem is one of the most important parts of an autonomous vehicle, therefore it has attracted substantial attention [7, 8]. It deals with searching a feasible path between the present location and the desired target while taking into consideration the geometry of the vehicle and its surroundings, its kinematic constraints and other factors that may affect the feasible path. Different methodologies are used to find feasible paths (see [9] for an overview). Some recent path-planning algorithms can be found in [10–12]. In [10] *Saska* et al. introduce a technique that integrates a spline-planning mechanism with a receding horizon control algorithm. This approach makes it possible to achieve a good performance in multi-robot systems. In [11] an offline path-planning algorithm for UAVs in complex terrain is presented. The authors propose an algorithm which can be divided into two steps: firstly a probabilistic method is applied for local obstacle avoidance and secondly a heuristic search algorithm is used to plan a global trajectory. In [12] *Zhang* et al. present a guidance principle for the path-following control of underactuated ships. They propose to split the path into regular straight lines and smooth arcs, using a virtual guidance ship to obtain the control input references that the real ship should have in order to follow the computed path. As it can be seen, there are many methods to obtain feasible paths for UVs; however, most of them do not consider the dynamics of the UV that should follow the path.

In their recent review article [13], *Yang* et al. have surveyed different path-planning algorithms. The authors discuss the fundamentals of the most successful robot 3D path-planning algorithms that have been developed in recent years. They mainly analyze algorithms that can be implemented in aerial robots, ground robots and underwater robots. They classify the different algorithms into five categories: i) sampling based algorithms, ii) node based algorithms, iii) mathematical model based algorithms (which include optimal control and receding horizon strategies), iv) bioinspired algorithms, and v) multifusion based algorithms. From these, only mathematical model based algorithms are able to incorporate in a simple way both the environment (kinematic constraints) and the vehicle dynamics in the path-planning process. Recently, in [14] *Hehn and D'Andrea* introduced a trajectory generation algorithm that can compute flight trajectories for quadcopters.

The proposed algorithm computes three separate translational trajectories (one for each degree of freedom) and guarantees the individual feasibility of these trajectories by deriving decoupled constraints through approximations. The authors do consider the quadcopter dynamics when they compute the flight trajectories but their proposed technique is not a general one (it can not be used with ground vehicles, for example). Even though the feasibility is guaranteed for each separate trajectory, the resulting vehicle trajectory might not be necessarily feasible (e.g., when perturbations are present). In [15] the authors present three conventional holonomic trajectory generation algorithms (flatness, polynomial and symmetric) for ground vehicles subject to constraints on their steering angle. In order to satisfy this constraint, they propose to lengthen the distance from the initial position to the final position until the constraint is satisfied. This process might be tedious and it may not be applicable in dynamic environments. Besides, it can only be used with ground vehicles and it can only handle steering constraints violations.

Motivated by the advent of new autonomous vehicles that encompass a broad range of mission capabilities, a suitable path-planning algorithm should be practicable and tailored to various UVs when executed in dynamical environments. Therefore, a challenging idea for path-planning is to develop an algorithm capable of handling dynamical environments and UVs that have different characteristics with regard to kinematic properties and maneuverability. For example, an autonomous rotary-wing vehicle is able to stop and make quick turns on a spot. On the contrary, an autonomous fixed-wing aircraft has to maintain a minimal flight velocity and can not turn at a large angle instantaneously. If a path obtained from a planning algorithm demands many agile or abrupt maneuvers, it would be difficult or even completely impossible to track. Consequently, it is inadequate in practice for a planning algorithm to only aim at an invariable model of steady maneuver.

In this article a unified framework to design an online path-planning algorithm is presented. The proposed strategy can be summarized in Fig. 1. Using a simplified

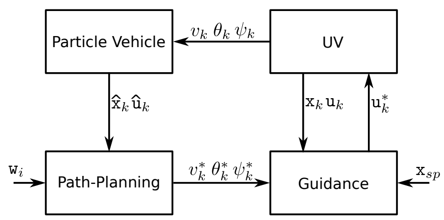


Fig. 1 Scheme of the path-planning & guidance system

particle vehicle (PV) model, which is configured to have similar characteristics (states and inputs constraints) to the UV, the path-planning module computes the velocity vector $\mathbf{v}_k^*$ (magnitude v_k^* and angles θ_k^* and ψ_k^*) in order to find the shortest feasible path towards the nearest waypoint $\mathbf{w}_i$. The vector $\mathbf{v}_k^*$ is in fact the velocity vector that the UV should have in order to achieve $\mathbf{w}_i$. Thus, using this velocity vector and other possible setpoints $\mathbf{x}_{sp}$, the guidance module is able to compute the inputs (actuator positions and motors speeds) that the UV should have so as to move towards $\mathbf{w}_i$. In this article, we mainly focus on the design of the path-planning module. We propose to design this module using the iterated robust NMPC technique presented in [16] as it uses a successive linearization method which allows us to use analytic tools to evaluate stability, robustness and convergence issues. Besides, it allows us to use quadratic program (QP) solvers and to easily take into account dynamic and physical constraints of the UV at the path-planning stage in order to obtain feasible paths.

The main contribution of this paper are: i) the proposal of a general algorithm for path-planning that can be used with any kind of UV, ii) the inclusion of the dynamics and constraints of the UV in the path-planning problem, iii) the guarantee of feasibility of the computed optimal path, iv) the inclusion of static and dynamic obstacles into the path planning problem, and v) the decentralization of the path-planning problem for multiple vehicles.

The organization of this article is as follows: in section 2 the 2D and 3D PV models are presented. In section 3, the path-planning problem is introduced. In section 4 three simulation examples are outlined. Finally, in section 5 conclusions are presented.

2 Non-linear Particle Vehicle Model

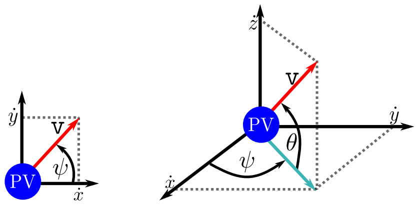
In this work we propose to use a PV model to obtain feasible and optimal paths for UVs. This section is devoted to obtain such a model for both the 2D and 3D cases. First, we provide a more general approach about systems representation and then we particularize it for the case of 2D and 3D PV models.

The general representation of the dynamics of an arbitrary non-linear system is given by

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)), \tag{1}$$

where $\mathbf{x}(t) \in \mathcal{X} \subseteq \Re^n$, $\mathbf{u}(t) \in \mathcal{U} \subseteq \Re^m$ and $\mathbf{d}(t) \in \mathcal{D} \subseteq \Re^v$ are the model state, input and disturbance vectors, respectively; $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{D}$ are the state, input and disturbance constraint sets; $f(\cdot)$ is a continuous and twice differentiable vector function that depends on the system being modeled[1].

To obtain the PV models, we use the 2D and 3D schemes shown in Fig. 2. Using these schemes, we propose to use the following state vector to model the 2D



(a) 2D **(b)** 3D

Fig. 2 Schemes of the proposed PV models

PV

$$\mathbf{x} = [x,\, y,\, v]^T, \tag{2}$$

where x and y denote the PV position coordinates and v is the modulus of the PV velocity vector. We define the control input vector as

$$\mathbf{u} = [\psi,\, \mathcal{T}]^T, \tag{3}$$

where ψ and $\mathcal{T}$ denote the yaw angle and the thrust force, respectively. Consequently, the 2D dynamics of the proposed PV model can be obtained as

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{d}) = \begin{bmatrix} v\cos\psi + d_x \\ v\sin\psi + d_y \\ -\tau v + \kappa\mathcal{T} \end{bmatrix}, \tag{4}$$

[1] To simplify the notation, from now on we will omit the time dependence, i.e. $\dot{\mathbf{x}}(t) = \dot{\mathbf{x}}$, $\mathbf{x}(t) = \mathbf{x}$, $\mathbf{u}(t) = \mathbf{u}$ and $\mathbf{d}(t) = \mathbf{d}$

where d_x and d_y are the xy components of $\mathbf{d}$, the damping constant τ determines the rate of change of the PV velocity and κ is a constant proportional to the thrust force $\mathcal{T}$.

To model the 3D PV we just have to include the altitude dependence. Using the scheme presented in Fig. 2b, the state vector is chosen as

$$\mathbf{x} = [x,\, y,\, z,\, v]^T, \tag{5}$$

where x, y and z denote the PV position coordinates and v is the modulus of the PV velocity vector. The control input vector is then defined as

$$\mathbf{u} = [\theta,\, \psi,\, \mathcal{T}]^T, \tag{6}$$

where θ, ψ and $\mathcal{T}$ denote, respectively, the pitch angle, the yaw angle and the thrust force. Then, the 3D dynamics of the PV model can be described by the following first order differential equation system:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{d}) = \begin{bmatrix} v\cos\theta\cos\psi + d_x \\ v\cos\theta\sin\psi + d_y \\ v\sin\theta + d_z \\ -\tau v + \kappa\mathcal{T} \end{bmatrix}, \tag{7}$$

where d_x, d_y and d_z are the xyz components of $\mathbf{d}$. As it can be seen, if the pitch angle θ is zero, then (7) is reduced to (4).

One important thing we would like to mention about the proposed PV models is that in the last equation of (4) and (7) the basic dynamics of the UV is included. This is very advantageous as physical systems do not have the ability to make instant changes in their dynamics. So, by including this last equation in the PV models we ensure that if this model is used in the path-planning module, then the path will be computed reflecting the UV basic dynamics, and consequently guaranteeing the feasibility of the path. Generally, the UV dynamics is not taken into account in path-planning algorithms because they use impulsional models [12, 17], which can lead to unfeasible paths for a UV.

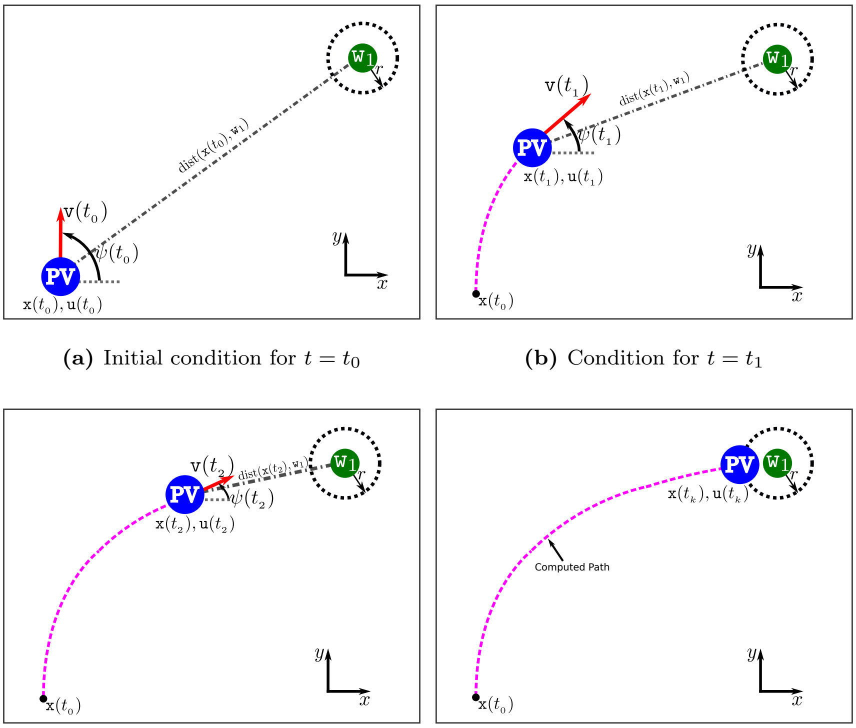
3 The Path-Planning Problem

Given a target position or *waypoint* $\mathbf{w}_i$ the path-planning problem consists in finding a path that connects the initial state vector $\mathbf{x}(t_0)$ and each consecutive waypoint $\mathbf{w}_i$[2], where the subscript $i = 1, 2, \cdots, M$ indicates the waypoint number. In this article we propose to find the path that is not only the shortest one but also a feasible one, i.e. the shortest path that also takes into account the dynamics and physical constraints of the UV that should follow the path. To find the shortest path we only have to measure the distance between the current position of the PV and the desired waypoint, and then minimize it. But as we also want the path to be feasible, we have to include the dynamics and constraints in the minimization problem. This may be done, for example, using a receding horizon technique, since the distance can be embedded in the cost function and the dynamics and constraints in the constrained minimization. Here, we propose to use the NMPC technique presented in [16] to control the velocity vector (modulus and direction) of the PV model. By controlling this vector the position of the PV is actually determined, thus defining the desired path towards the waypoint. The main advantage of using this technique (unlike the one used in [12], for example)

[2] For the 2D case $\mathbf{w}_i$ is defined as $\mathbf{w}_i = [w_{i_x}, w_{i_y}, w_{i_v}]^T$ and for the 3D case $\mathbf{w}_i = [w_{i_x}, w_{i_y}, w_{i_z}, w_{i_v}]^T$. w_{i_x}, w_{i_y} and w_{i_z} denote the xyz coordinates of waypoint $\mathbf{w}_i$ and w_{i_v} defines the speed that the PV should have when $\mathbf{w}_i$ is reached.

is that, as the dynamics and constraints of the UV that should follow the path can be taken into account in the minimization problem, then the resulting path is guaranteed to be feasible.

In Fig. 3 a scheme of the proposed methodology is shown. Under the assump-



(a) Initial condition for $t = t_0$ **(b)** Condition for $t = t_1$

(c) Condition for $t = t_2$ **(d)** Condition for $t = t_k$

Fig. 3 Computing a path between $\mathbf{x}(t_0)$ and $\mathbf{w}_1$

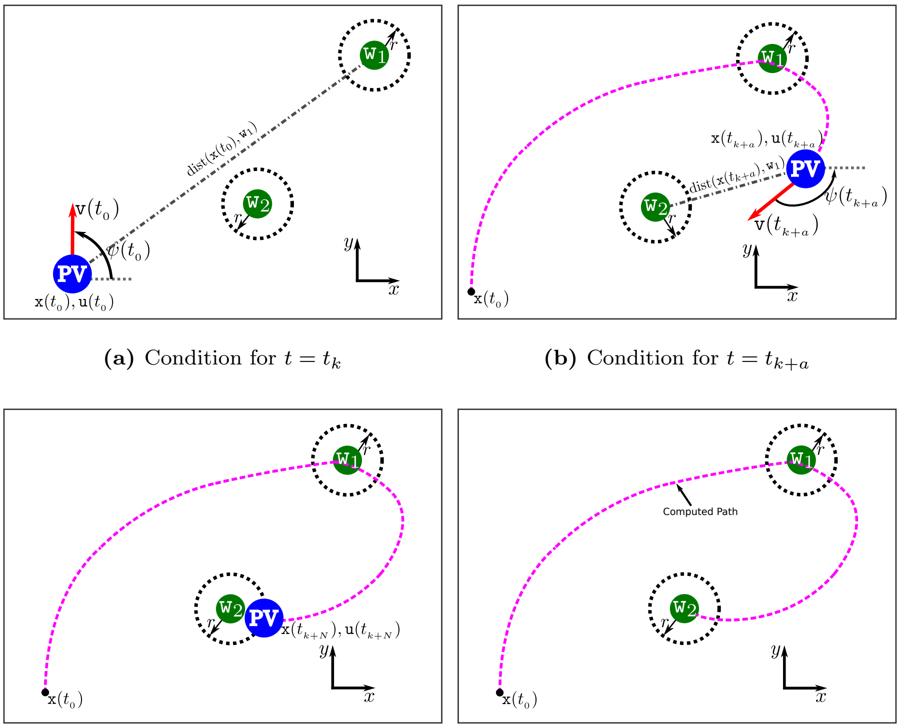
tion that the control inputs of the PV have a limited rate of change, this figure shows how the path towards a single waypoint $\mathbf{w}_1$ is obtained. As can be seen in Figs. 3b and 3c, the PV starts moving towards $\mathbf{w}_1$. To do this, we propose to use the algorithm [16] to minimize the euclidean distance $(\text{dist}(\mathbf{x}(t_j), \mathbf{w}_i))$ between the current position of the PV and the desired waypoint. As a result, the optimal yaw angle and thrust force are computed and the velocity vector modifies its direction and modulus in order to reach the desired target in a feasible way. The path we were looking for turns out to be the path that the PV has described in order to go from the starting configuration to the desired one (see Fig. 3d).

Also, it could happen that a path that connects the initial position and several waypoints is required. This situation is illustrated in Fig. 4 for the case of two waypoints[3]. As shown in Fig. 4a, the PV is configured with an initial condition $\mathbf{x}(t_0)$, $\mathbf{u}(t_0)$ and the path should pass first through the waypoint $\mathbf{w}_1$ and then through the waypoint $\mathbf{w}_2$. To obtain this path, two sub-paths are considered: one joining the initial configuration with $\mathbf{w}_1$ and the other joining $\mathbf{w}_1$ with $\mathbf{w}_2$. The first sub-path is obtained in a similar way as we have done in Fig. 3. Once $\mathbf{w}_1$ has been reached, the second sub-path can be computed. To do this, the desired target is changed from $\mathbf{w}_1$ to $\mathbf{w}_2$ and the minimization of the distance between the current position of the PV and $\mathbf{w}_2$ is performed. As a result, the PV starts moving again and its velocity vector is recalculated in order to move the PV towards $\mathbf{w}_2$ (see Figs. 4b and 4c). The full computed path can be seen in Fig. 4d. As it is shown, it has been obtained by joining both sub-paths together.

As was mentioned before, we propose to modify the direction and modulus of the velocity vector of the PV using the control technique described in [16]. To use this control technique, first we need to transform the non-linear model (1) into an equivalent discrete linear time-varying (LTV) one of the form

$$\tilde{\mathbf{x}}_{k+1|k} = \mathbf{A}_{k|k}\tilde{\mathbf{x}}_{k|k} + \mathbf{B}_{u_{k|k}}\tilde{\mathbf{u}}_{k|k} + \mathbf{B}_{d_{k|k}}\tilde{\mathbf{d}}_{k|k}, \tag{8}$$

[3] Note that if there are more than two waypoints, the procedure to compute the path is similar as the one presented here.



(a) Condition for $t = t_k$

(b) Condition for $t = t_{k+a}$

(c) Condition for $t = t_N$

(d) Full computed path

Fig. 4 Computing a path between $\mathbf{x}(t_0)$ and $\mathbf{w}_2$ passing through $\mathbf{w}_1$

where

$$\tilde{\mathbf{x}}_{k|k} = \mathbf{x}_{k|k} - \mathbf{x}^r_{k|k}, \quad \tilde{\mathbf{u}}_{k|k} = \mathbf{u}_{k|k} - \mathbf{u}^r_{k|k} \quad \text{and} \quad \tilde{\mathbf{d}}_{k|k} = \mathbf{d}_{k|k} - \mathbf{d}^r_{k|k}. \tag{9}$$

$\mathbf{x}^r_{k|k}$, $\mathbf{u}^r_{k|k}$ and $\mathbf{d}^r_{k|k}$[4] define the linearization state, input and disturbance trajectories, and $\mathbf{A}_{k|k}$, $\mathbf{B}_{u_{k|k}}$ and $\mathbf{B}_{d_{k|k}}$ are the discrete matrices of the linearized version of (1). Receding horizon techniques use a cost function of the form

$$\mathcal{J}(k) = \sum_{j=0}^{N-1} \mathcal{L}_j(\mathbf{x}_{k+j|k}, \mathbf{u}_{k+j|k}) + \mathcal{L}_N(\mathbf{x}_{k+N|k}), \tag{10}$$

[4] $\mathbf{d}^r_{k+j|k}$, $j = 0, \cdots, N-1$ is a given or estimated perturbation

where $\mathcal{L}_j(\cdot,\cdot)$ is the stage cost and $\mathcal{L}_N(\cdot,\cdot)$ stands for the terminal cost. Generally, in receding horizon algorithms both the stage cost and the terminal cost are adopted as follows:

$$\mathcal{L}_j(\mathbf{x}_{k+j|k}, \mathbf{u}_{k+j|k}) = \|\mathbf{x}_{k+j|k} - \mathbf{w}_i\|^l_{\mathbf{Q}_{k|k}} + \|\Delta\mathbf{u}_{k+j|k}\|^p_{\mathbf{R}_{k|k}} \tag{11}$$

and

$$\mathcal{L}_N(\mathbf{x}_{k+N|k}) = \|\mathbf{x}_{k+N|k} - \mathbf{w}_i\|^l_{\mathbf{P}_{k|k}}, \tag{12}$$

where $\mathbf{Q}_{k|k}, \mathbf{R}_{k|k}, \mathbf{P}_{k|k}$ are positive definite matrices; $\mathbf{P}_{k|k}$ is the terminal weight matrix that is chosen so as to satisfy the Lyapunov equation. Superscripts l and p are even positive numbers and in general are adopted as $l = p = 2$. $\|(\cdot)\|^\beta_\alpha$ stands for the α-weighted β-norm and $\Delta\mathbf{u}_{k+j|k} = \mathbf{u}_{k+j|k} - \mathbf{u}_{k+j-1|k}$. Then, in terms of the LTV system (8) and according to [16], we propose to solve the following optimization problem:

$$\begin{gathered}\min_{\tilde{\mathbf{U}}_k \in \mathcal{U}} \ \mathcal{J}(k) \\ \text{st.} \begin{cases} \tilde{\mathbf{x}}_{k+j|k} = \mathbf{A}_{k|k}\tilde{\mathbf{x}}_{k|k} + \mathbf{B}_{k|k}\tilde{\mathbf{u}}_{k|k}, \\ \tilde{\mathbf{x}}_{k|k} = \mathbf{x}_{k|k} - \mathbf{x}^r_{k|k}, \\ \tilde{\mathbf{u}}_{k|k} = \mathbf{u}_{k|k} - \mathbf{u}^r_{k|k}, \\ \mathcal{J}(k) \leq \mathcal{J}_0(k). \end{cases}\end{gathered} \tag{13}$$

where

$$\tilde{\mathbf{U}}_k{}^5 = [\tilde{\mathbf{u}}_{k|k},\ \tilde{\mathbf{u}}_{k+1|k},\ \cdots\ \tilde{\mathbf{u}}_{k+N-1|k}]^T \tag{14}$$

is the control input sequence and $\mathcal{J}_0(k)$ denotes the cost function evaluated for the initial solution $\mathbf{U}^0_k = [\mathbf{u}_{k|k-1}, \mathbf{u}_{k+1|k-1}, \cdots, \mathbf{u}_{k+N-2|k-1}, 0]^T$ at iteration k.

[5] Hereinafter we use bold capital fonts to denote complete sequences computed for $k, k+1, \cdots, k+N-1$.

The last inequality in (13) defines the contractive constraint that guarantees the convergence of the iterative solution and defines an upper bound for $\mathcal{J}(k)$ (for a detailed explanation about this contractive constraint, please refer to [16]).

In this work, we propose to split the optimization problem (13) into two different ones. When the current position of the PV is far away from the desired waypoint (that is to say when dist[6]$(\mathbf{x}_{k|k}, \mathbf{w}_i) > r$) we use the standard quadratic objective function, i.e. we set $l = 2$ and $p = 2$ [7]. Instead, when the current position of the PV is sufficiently close to the desired waypoint (say $\mathrm{dist}(\mathbf{x}_{k|k}, \mathbf{w}_i) \leq r$) we adopt a higher order objective function, i.e we set $l = 4$ and $p = 2$.

Remark 1 Note that with the selected values of l and p both optimization problems are convex, thus not affecting the optimality, feasibility and stability of the system.

The main idea behind changing the order of the optimization problem can be explained looking at Fig. 5. This figure shows how the one dimensional (1D) cost function

$$c(e) = \left(\frac{e - w_i}{r}\right)^l \tag{15}$$

changes its shape as the parameter l is varied. As it can be seen, when we are sufficiently close to the desired waypoint, as l gets bigger the derivative of the cost

[6] $\mathrm{dist}(a, b)$ is a function that computes the euclidean distance between a and b. r is the radius of a circumference (2D case) or a sphere (3D case) centred at $\mathbf{w}_i$ that defines the zone in which we consider that we are close enough to the waypoint and proceed to follow the next one.

[7] Note that with $l = 2$ the position error for both the stage cost and the terminal cost are the square of the distance between the current position and the desired one, which is the quantity we want to minimize. $p = 2$ was selected in order to obtain a quadratic function for input variations. Also, with the proposed superscripts selection, the cost function can be seen as a measure of the energy expended in the path-planning process.

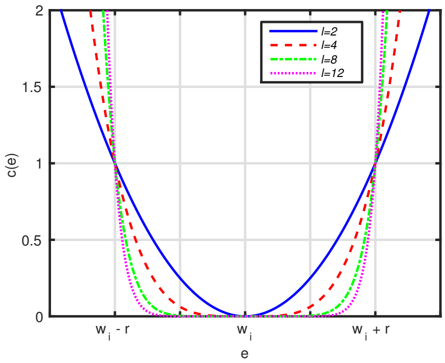


Fig. 5 One dimensional cost function

function around $e = w_i \pm r$ is very sharp. This effect can be used to force the PV to reach the desired waypoint faster than if the conventional quadratic function ($l = 2$) is used. For the multi-dimensional case, the effect of varying l is similar as the one described for the 1D case. We have discussed the 1D case because it can be visualized graphically.

The proposed path-planning algorithm can be described as follows: first, the non-linear system is transformed into a LTV system by means of successive linearizations along pre-defined state-space trajectories. Then, the distance between the current position of the PV and the desired waypoint is measured. If the PV is close enough to the target, then the higher order optimization problem is adopted to force the PV to reach the waypoint faster than if the conventional optimization problem is used. The proposed constrained minimization problem is solved and the optimal control input sequence (orientation angles and thrust force) is then obtained. The minimization process is repeated until the control input sequence converges. Finally, the computed optimal inputs are applied to the PV and the path-planning process is reinitialized. In Algorithm 1, the proposed path-planning algorithm is summarized.

Remark 2 Note that the change in the objective function does not affect the convergence and stability because $\mathcal{J}(k) \leq \mathcal{J}_0(k)$ guarantees these properties for time-variant objective functions [16].

Remark 3 If multiple vehicles are considered in the path-planning procedure, the proposed algorithm can be used in a decentralized manner because the couplings between vehicles can be embedded into the objective function and the constraints of the minimization problem [18].

4 Simulation Examples

In this section several simulation examples are shown. Using the PV model (4) we solve the optimization problem (13) to find 2D feasible and optimal paths. The extension to the 3D case is straightforward, we only need to use the PV model (7) instead of (4).

For all the simulation examples we assumed that there are no disturbances ($\mathbf{d}_{k|k} = 0$) and that the PV has the initial state vector $\mathbf{x}_0 = [0,\,0,\,0]^T$ and the initial input vector is $\mathbf{u}_0 = [\pi/2,\,0]^T$. The PV model is discretized using a sampling rate $T_s = 0.1\,\mathrm{s}$ and the horizon N was set to $N = 8$. The input weight matrix is chosen as $\mathbf{R}_{k|k} = diag([0.1,\,0.1])$. The PV constraints are configured as follows: $0 \leq \mathcal{T} \leq 2\,(\mathrm{N})$, $-0.087 \leq \Delta\psi \leq 0.087\,(\mathrm{rad/s})$, $-1 \leq \Delta\mathcal{T} \leq 1\,(\mathrm{N/s})$ and $0 \leq v \leq 2\,(\mathrm{m/s})$. ψ, x and y are unconstrained. Both constants of the PV model are set as $\tau = 2\,(1/\mathrm{s})$ and $\kappa = 2\,(1/\mathrm{kg})$. As we are interested in having the computed path

Algorithm 1: The Path-planning algorithm

Require: The initial condition $\mathbf{x}_{k|k}$, the iteration index $q = 0$ and the PV models (4) or (7)

1: Obtain the LTV system $[\mathbf{A}_{k|k}, \mathbf{B}_{k|k}]$ and the matrices $Q^q_{k|k}$, $R^q_{k|k}$ and $P^q_{k|k}$.
2: **if** $\mathrm{dist}(\mathbf{x}_{k|k}, \mathbf{w}_i) > r$ **then**
3: $\quad l \leftarrow 2$
4: $\quad p \leftarrow 2$
5: **else**
6: $\quad l \leftarrow 4$
7: $\quad p \leftarrow 2$
8: **end if**
9: Compute the optimal control input sequence $\tilde{\mathbf{U}}^{*,q}_k$ solving (13)
10: Update $\mathbf{U}^{*,q}_{k|k} \leftarrow \mathbf{U}^{r,q}_{k|k} + \tilde{\mathbf{U}}^{*,q}_{k|k}$
11: **if** $\left\| \mathbf{U}^{*,q}_k - \mathbf{U}^{*,q-1}_k \right\|_\infty \leq \epsilon$ **then**
12: $\quad \mathbf{U}^*_k \leftarrow \mathbf{U}^{*,q}_k$
13: $\quad k \leftarrow k + 1$
14: $\quad q \leftarrow 0$
15: **else**
16: $\quad q \leftarrow q + 1$
17: $\quad$ Update $\mathbf{U}^q_k = \mathbf{U}^{*,q-1}_k$
18: $\quad$ Go back to line 2
19: **end if**
20: Apply $\mathbf{u}_{k|k} = \mathbf{u}^*_{k|k}$ to the system
21: Go back to line 1

pass sufficiently close to the waypoints, but not exactly through them, we define a circular area centered at each waypoint. If the path passes through this area, then we consider that the corresponding waypoint has been reached. For all the waypoints we set this area to a disk with a radius of $r = 0.4\,(m)$.

In examples 1 and 2 we assume that the computed path should pass sufficiently close to the following three waypoints[8]:

$$\begin{aligned} \mathbf{w}_1 &= [-10,\, 0,\, 1]^T, \\ \mathbf{w}_2 &= [3,\, 8,\, 1]^T, \\ \mathbf{w}_3 &= [-2,\, -5,\, 0]^T. \end{aligned} \tag{16}$$

For these examples, the weight matrices are adopted as follows:

1. Between the initial position $\mathbf{x}_0$ and the first waypoint $\mathbf{w}_1$, the state weight matrix is adopted as $\mathbf{Q}_{k|k} = diag([10,\, 10,\, 10])$.
2. Between the waypoints $\mathbf{w}_1$ and $\mathbf{w}_2$, we adopt $\mathbf{Q}_{k|k} = diag([10,\, 10,\, 100])$.
3. Between the waypoints $\mathbf{w}_2$ and $\mathbf{w3}$, the weight matrix is adopted as $\mathbf{Q}_{k|k} = diag([10,\, 10,\, 100])$.

In example 3 we assume that the computed path should pass sufficiently close to the following waypoint:

$$\mathbf{w}_1 = [10,\, 8,\, 0]^T. \tag{17}$$

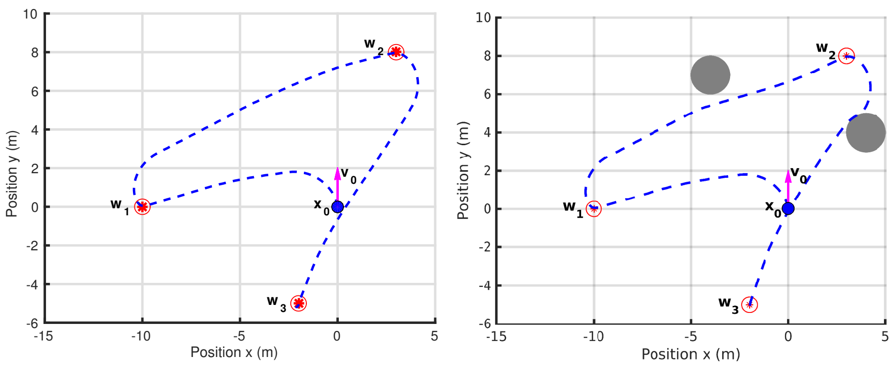
For this example, the weight matrix is adopted as $\mathbf{Q}_{k|k} = diag([10,\, 10,\, 10])$.

At this point it is worth mentioning that with the proposed approach all the examples could be run in real-time, as the optimization loop was solved in a maximum time of approximately 30 (ms) in a desktop PC (i7-2600K CPU@3.40GHz, 32GB RAM). All the examples were programmed using Python and CasADi [19] and were not optimized in any way. Since CasADi has C++ interfaces, we think that there is still room for improvement.

[8] As the simulation examples are performed for the 2D case, the components of the following waypoints denote, respectively, x-coordinate, y-coordinate and speed.

4.1 Example 1

In this example we use the PV model (4) to compute a feasible and optimal path that passes sufficiently close to the waypoints $\mathbf{w}_i$ $(i = 1, 2, 3)$ defined in (16). We consider two cases: i) There are no obstacles between the waypoints, and ii) There are two circular obstacles of radii $r_{o_1} = r_{o_2} = 1\,(\text{m})$ and centers $\mathbf{c}_{o_1} = [-4,\ 7]^T\,(\text{m})$ and $\mathbf{c}_{o_2} = [4,\ 4]^T\,(\text{m})$. In the first case, the computed path only has to satisfy the constraints imposed by the dynamics of the PV. In the second one, the computed path not only has to satisfy the PV constraints but it also has to avoid colliding with the two predefined obstacles. The results are shown in Fig. 6. As can be seen,



(a) Without obstacles **(b)** With two obstacles

Fig. 6 Computed paths using the path-planning algorithm

in both cases the PV starts from $\mathbf{x}_0$ with a velocity vector $\mathbf{v}_0$. When there are no obstacles (Fig. 6a), the resulting path is smooth and it satisfies the PV state and input constraints. Moreover, the PV passes successfully through the three desired waypoints. When the two circular obstacles are present (Fig. 6b) the path is similar to that obtained in the previous case but in this situation the computed path avoids collision with both obstacles. With the proposed methodology, adding the obstacles is just as simple as adding constraints of the form

$$(x - c_{o_{i_x}})^2 + (y - c_{o_{i_y}})^2 \geq r_o \tag{18}$$

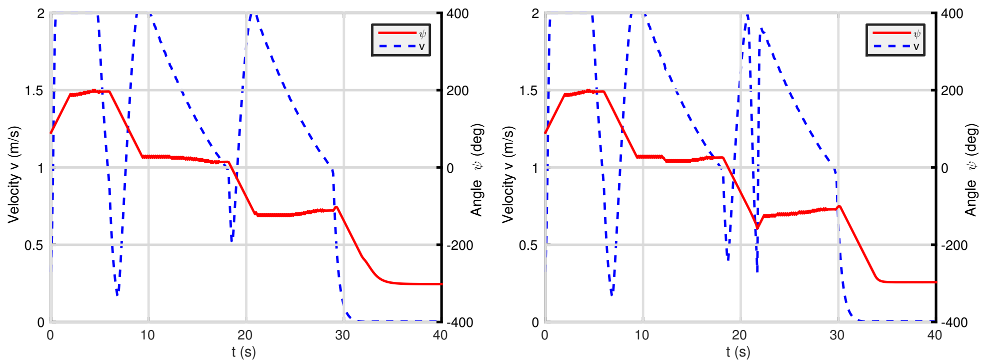
to the optimization problem (13), where $c_{o_{i_x}}$ and $c_{o_{i_y}}$ denote the x and y components of the vector $\mathbf{c}_{o_i}$. Since the obstacles are added as constraints to (13), their detection and avoidance is straightforward, because the solution of the optimization problem already takes into account the presence of these static obstacles.

Remark 4 Note that any obstacle can be circumscribed within a circle (2D) or a sphere (3D), thus any shape of obstacle can be considered.

Fig. 7 shows the evolution of the velocity modulus and the yaw angle of the PV for both situations, without obstacles (Fig. 7a) and with them (Fig. 7b). Both figures are similar, the major difference that they exhibit can be observed approximately between $t = 21\,(\mathrm{s})$ and $t = 23\,(\mathrm{s})$. At this time the PV has reached the waypoint $\mathbf{w}_2$ and it is moving towards $\mathbf{w}_3$. When the PV detects the presence of the obstacle centered at $\mathbf{c}_{o_2}$ it must reduce its velocity and modify its yaw angle in order to avoid colliding with the obstacle. Once the obstacle is avoided, the PV accelerates again and reaches the final target $\mathbf{w}_3$.

4.2 Example 2

For the second simulation example, we consider that the PV should pass sufficiently close to waypoints $\mathbf{w}_i$ $(i = 1, 2, 3)$ and that there are three obstacles: the previous two mentioned in example 1, and a third one that appears suddenly at $t = 2.5\,(\mathrm{s})$. The latter obstacle is also circular and it is located at $\mathbf{c}_{o_3} = [-6,\, 2]^T\,(\mathrm{m})$ with

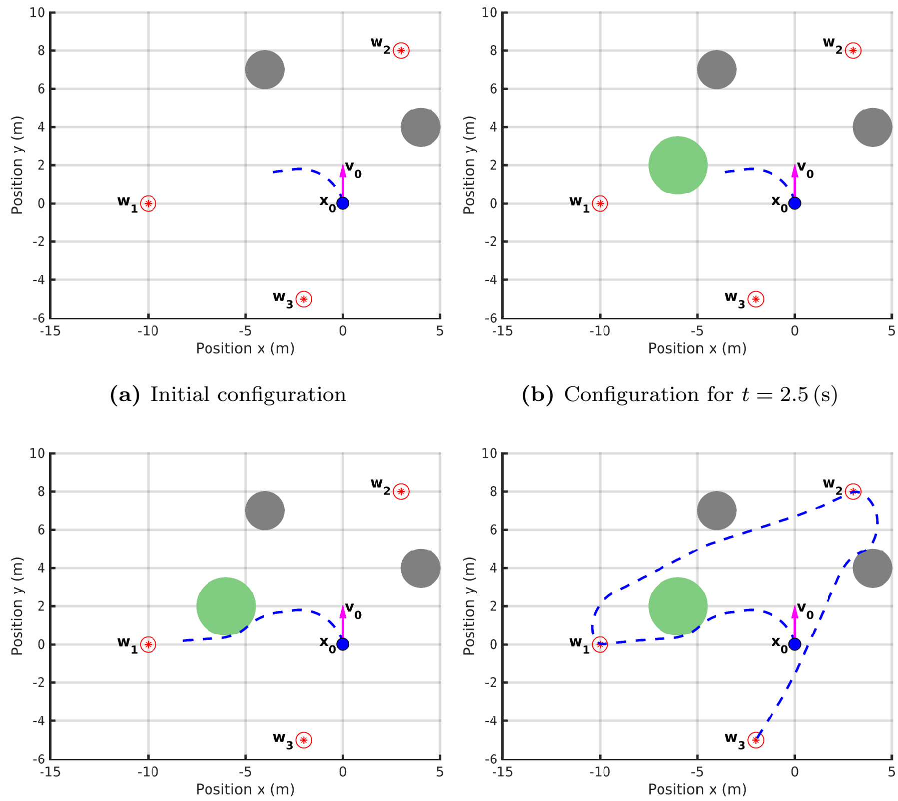


(a) Without obstacles

(b) With two circular obstacles

Fig. 7 Evolution of velocity modulus and orientation angle

radius $r_{o_3} = 1.5\,(\mathrm{m})$. The results are shown in Fig. 8. As observed in Fig. 8a, when the PV starts moving only two fixed obstacles are present. At $t = 2.5\,(\mathrm{s})$ (see Fig. 8b) a new circular obstacle appears in the way of the PV. If the PV does not change the direction of its velocity vector $\mathbf{v}$, then it will collide with this new obstacle. Fortunately, we are computing the path in an online manner, so, as it is shown in Fig. 8c, the PV can detect the new obstacle and the direction of the velocity vector is automatically changed. In Fig. 8d the resulting path is depicted, which is smooth and it satisfies not only the PV constraints but also avoids the two static obstacles and the appearing one. The evolution of the velocity modulus and the yaw angle is depicted in Fig. 9. As can be observed, a peak appears in the yaw angle curve between $t = 2.5\,(\mathrm{s})$ and $t = 4.5\,(\mathrm{s})$. This occurs because the PV needs to modify its yaw angle from $\psi = 180\,(\mathrm{deg})$ up to $\psi = 220\,(\mathrm{deg})$ in order to change its direction and, consequently, avoid colliding with the new obstacle. The evolution of the velocity modulus is similar to that presented in Fig. 7b.



(a) Initial configuration

(b) Configuration for $t = 2.5\,(\mathrm{s})$

(c) Configuration for $t = 5\,(\mathrm{s})$

(d) Final configuration

Fig. 8 Generation of a feasible path with two fixed obstacles and one appearing at $t = 2.5\,(\mathrm{s})$

4.3 Example 3

In this example we explore the problem of computing a feasible path in a follower condition without obstacles. We assume that we have two particle vehicles: PV_1, with initial condition $\mathbf{x}_{0_1} = [0,\, 0,\, 0]^T$ and $\mathbf{u}_{0_1} = [\pi/2,\, 0]^T$, and PV_2 (the follower) whose initial condition is $\mathbf{x}_{0_2} = [-5,\, 5,\, 0]^T$ and $\mathbf{u}_{0_2} = [-\pi/2,\, 0]^T$. To perform this simulation example, we configured two NMPC controllers: one for PV_1 and the

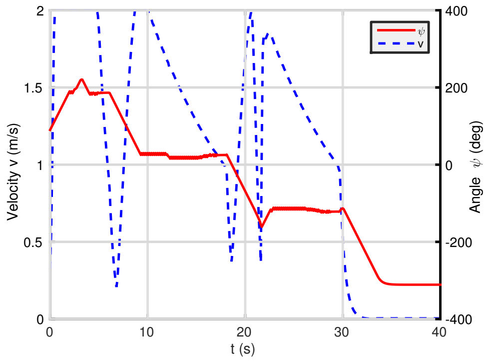
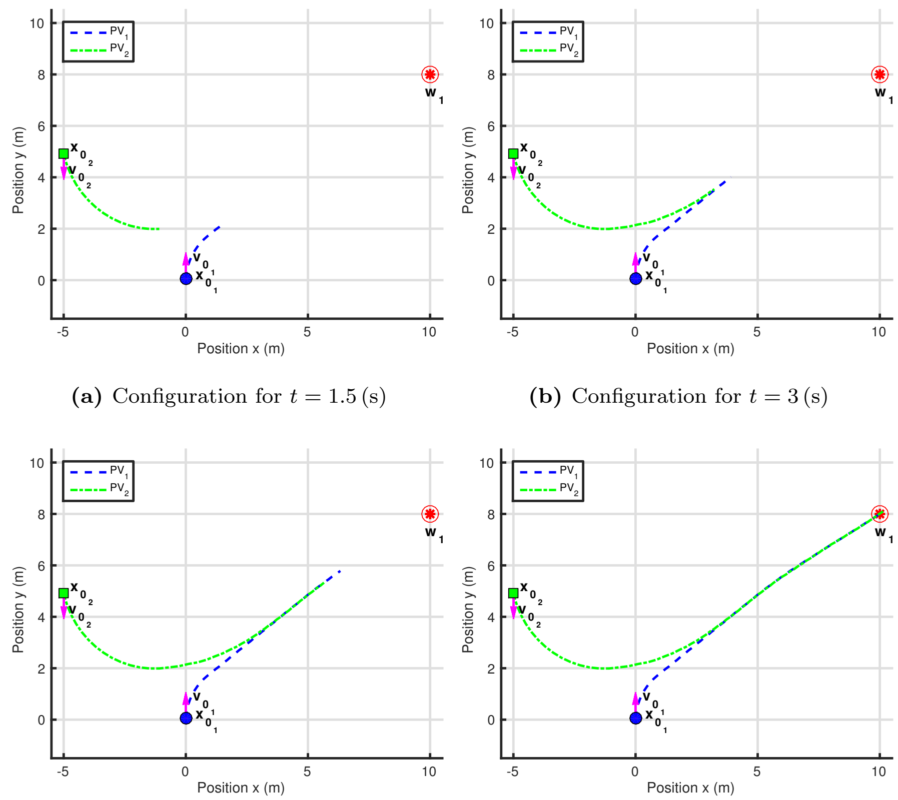


Fig. 9 Evolution of velocity modulus and orientation angle

other for PV_2. The configuration of the NMPC parameters for both controllers was done as described in the beginning of Section 4, except that we have let PV_2 to increment its speed up to $4\,(\mathrm{m/s})$.

In this example, we consider that PV_1 has to reach the waypoint (17) and PV_2 has to reach PV_1, i.e. its target is the current position of PV_1. The results obtained are shown in Fig. 10. As can be seen in Fig. 10a, at $t = 1.5\,(\mathrm{s})$ the PV_1 (rounded) starts moving towards $\mathbf{w}_1$ while the PV_2 (squared) modifies its velocity vector in order to move towards PV_1. At $t = 3\,(\mathrm{s})$ (Fig. 10b), the PV_2 is located behind the PV_1. When $t = 4.5\,(\mathrm{s})$ the PV_2 has almost reached the PV_1 and they both move towards $\mathbf{w}_1$, as depicted in Fig. 10c. Finally, as shown in Fig. 10d, when $t = 10\,(\mathrm{s})$ the PV_2 reaches PV_1 and they both reach the desired target $\mathbf{w}_1$. Figure 11 compares the velocities (Fig. 11a) and the yaw angles (Fig. 11b) of both PV_1 and PV_2. As it can be seen from Fig. 11a, the PV_2 increments its speed up to its maximum value $(4\,\mathrm{m/s})$ in order to follow as quick as possible the PV_1. Once the follower is close to PV_1, both velocities profiles are similar. In Fig. 11b it can be seen how both yaw angles are modified. At $t = 3\,(\mathrm{s})$ both ψ_1 and ψ_2 have similar



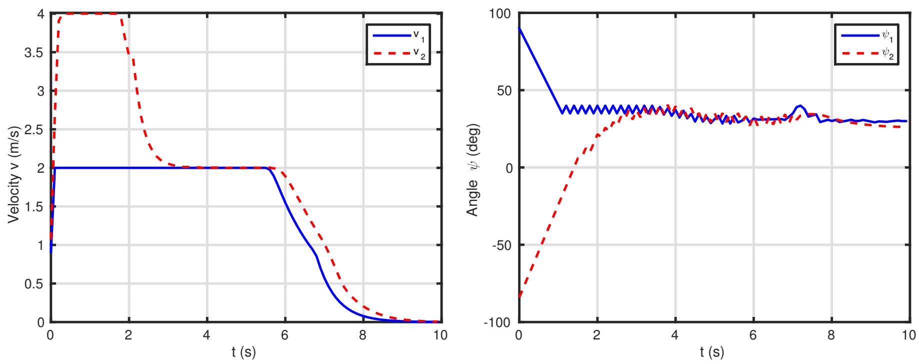
(a) Configuration for $t = 1.5\,(\mathrm{s})$ **(b)** Configuration for $t = 3\,(\mathrm{s})$

(c) Configuration for $t = 4.5\,(\mathrm{s})$ **(d)** Configuration for $t = 10\,(\mathrm{s})$

Fig. 10 Generation of a feasible path in a follower condition without obstacles

values, meaning that the velocity vectors of PV_1 and PV_2 are aligned with each other and consequently, PV_1 reaches the desired waypoint and the PV_2 reaches PV_1.

In this example we have computed a feasible path in a follower problem. If instead of following the moving object we configure PV_2 to move away from PV_1, then the proposed approach can be easily extended to be used with moving obstacles.



(a) Velocities of both PVs

(b) Orientation angle of both PV

Fig. 11 Evolution of the velocity modulus and orientation angles of both PVs without obstacles

4.4 Example 4

In this example we explore the problem of computing a feasible path while following it with a Husky[9] UGV simulated in Gazebo[10] simulator. The mathematical model used to solve the guidance of the Husky UGV is given by the following equation

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} v_h \cos \psi_h \\ v_h \sin \psi_h \\ \omega_h \end{bmatrix}, \tag{19}$$

where $\mathbf{x} = [x_h, y_h, \psi_h]^T$ is the state vector, x_h and y_h denote the robot position and ψ_h denotes its yaw angle. The control input vector $\mathbf{u} = [v_h, \omega_h]^T$ includes the linear and angular velocities v_h and ω_h, respectively. It should be noticed that the model (19) used to control the Husky is different from the plant, which is

[9] https://www.clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/

[10] http://gazebosim.org/

a complex model simulated by Gazebo. We assume that both the PV and the Husky UGV have the same initial position but they differ in its initial orientation angle. The PV has its velocity vector pointing upwards ($\psi = 90\,(\mathrm{deg})$) while the Husky velocity vector points to the east side ($\psi_h = 0\,(\mathrm{deg})$). As it can be seen in Fig. 12 the Husky model is able to follow the computed path successfully. This is mainly due to the fact that the navigation algorithm generates a trajectory that is feasible and takes into account the dynamic constraints of the UGV. Fig. 13a

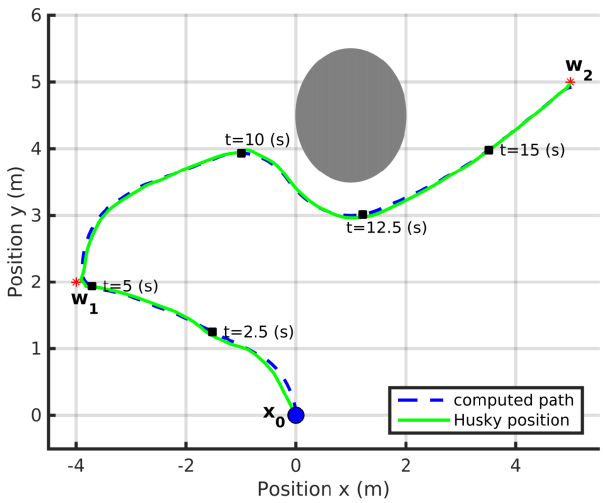
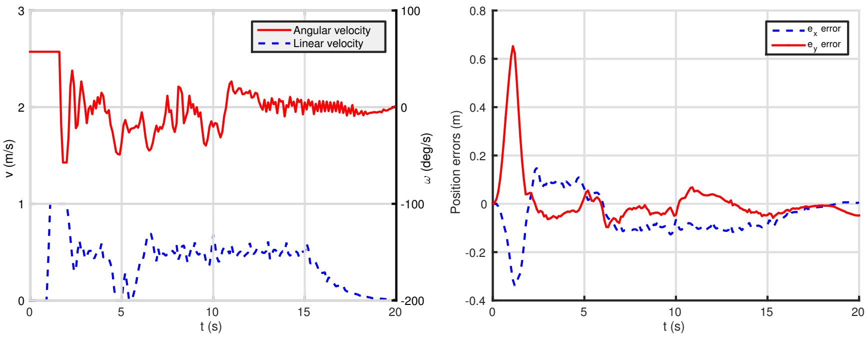


Fig. 12 Path-following with a Husky UGV

shows that when $t = 0\,(s)$, the Husky turns left in order to move towards waypoint $\mathbf{w}_1 = [-4,\ 2,\ 0.5]^T$. Once it is reached, the Husky UGV turns right in order to move in a straight line towards waypoint $\mathbf{w}_2 = [5,\ 5,\ 0]^T$. As the path was computed taking into account a circular obstacle which is located at $\mathbf{c}_o = [1,\ 4.5]^T\,(\mathrm{m})$ with radius $r_o = 1\,(\mathrm{m})$, it can be seen that the UGV performs the obstacle avoidance maneuver at approximately $t = 10\,(s)$ without any difficulty. After that, it moves towards waypoint $\mathbf{w}_2$ in almost a straight line. It should be emphasized that as the resulting path is smooth and feasible, the Husky is able to follow it without major difficulties. In Fig. 13b it is shown the xy-position errors $e_x = x - x_h$ and $e_y = y - y_h$, respectively, in the path-following maneuver. It can be seen that at approximately $t = 1\,(\mathrm{s})$ both errors tend to increase. This is due to the fact that the orientation of the Husky is different from the orientation of the PV. This was done on purpose in order to show that despite both yaw angles were different, the Husky is able to follow quite well the computed path. Then, for approximately $t > 2.5\,(\mathrm{s})$ the position errors tend to decrease and are very close to 0.



(a) Control inputs **(b)** Position errors

Fig. 13 Husky control inputs and path-following errors

5 Conclusions

In this article we have presented an online path-planning algorithm that can be used to guide any kind of unmanned vehicle towards desired targets. The proposed algorithm handles the problem of finding the optimal path towards desired waypoints, while taking into account the kinematic, the dynamic and constraint of the vehicle. We used a simplified particle vehicle model and the iterated non-linear model predictive control technique to control the velocity vector of this particle vehicle model. By controlling this vector we have actually determined the path that the particle vehicle model should take in order to reach the targets. We have also exploited the use of a higher order cost function in the optimization problem. Because we have used the iterated NMPC algorithm [16], optimality, stability and feasibility can be guaranteed. The performance and capabilities of the proposed path-planning algorithm were demonstrated through several simulation examples. The path-following capabilities were explored using a Husky UGV to follow a feasible path. All the simulation examples were performed successfully.

Acknowledgements The authors wish to thank the *Universidad Nacional de Litoral* (with CAI+D Jóven 500 201501 00050 LI and CAI+D 504 201501 00098 LI), the *Agencia Nacional de Promoción Científica y Tecnológica* (with PICT 2016-0651) and the *Consejo Nacional de Investigaciones Científicas y Técnicas* (CONICET) from Argentina, for their support.

References

1. P. Doherty and P. Rudol, “A uav search and rescue scenario with human body detection and geolocalization,” *Lecture Notes in Computer Science*, pp. 1–13, 2007.
2. G. E. Dewi Jones & Ian Golightly, Jonathan Roberts & Kane Usher, “Power line inspection-a UAV concept,” in *IEE Forum on Autonomous Systems*, no. November, pp. 2–7, 2005.

3. C. Zhang and J. Kovacs, “The application of small unmanned aerial systems for precision agriculture: a review,” 2012.
4. S. M. Adams, M. L. Levitan, and C. J. Friedland, *High Resolution Imagery Collection Utilizing Unmanned Aerial Vehicles (UAVs) for Post-Disaster Studies*, ch. 67, pp. 777–793. American Society of Civil Engineers, 2013.
5. A. Bouhraoua, N. Merah, M. AlDajani, and M. ElShafei, “Design and implementation of an unmanned ground vehicle for security applications,” in *7th International Symposium on Mechatronics and its Applications (ISMA)*, pp. 1–6, April 2010.
6. M. YAĞIMLI and H. S. Varol, “Mine detecting gps-based unmanned ground vehicle,” in *4th International Conference on Recent Advances in Space Technologies*, pp. 303–306, 2009.
7. United States Air Force, “Unmanned aircraft systems flight plan 2009–2047,” *Headquarters Department of the Air Force, Washington DC*, 2009.
8. Weatherington, Dyke and Deputy, U, “Unmanned aircraft systems roadmap, 2005–2030,” *UAV Planning Task Force, OUSD (AT&L)*, 2005.
9. S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.
10. M. Saska, V. Spurn, and V. Vonsek, “Predictive control and stabilization of nonholonomic formations with integrated spline-path planning,” *Robotics and Autonomous Systems*, 2015.
11. Q. Xue, P. Cheng, and N. Cheng, “Offline path planning and online replanning of uavs in complex terrain,” in *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*, pp. 2287–2292, 2014.
12. G. Zhang and X. Zhang, “A novel DVS guidance principle and robust adaptive path-following control for underactuated ships using low frequency gain-

learning," *ISA Transactions*, vol. 56, pp. 75 – 85, 2015.

13. Yang, Liang and Qi, Juntong and Song, Dalei and Xiao, Jizhong and Han, Jianda and Xia, Yong, "Survey of Robot 3D Path Planning Algorithms," *Journal of Control Science and Engineering*, vol. 2016, 2016.
14. M. Hehn and R. D'Andrea, "Real-time trajectory generation for quadrocopters," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 877–892, 2015.
15. Minh, Vu Trieu and Pumwa, John, "Feasible path planning for autonomous vehicles," *Mathematical Problems in Engineering*, vol. 2014, 2014.
16. M. Murillo, G. Snchez, and L. Giovanini, "Iterated non-linear model predictive control based on tubes and contractive constraints," *ISA Transactions*, vol. 62, pp. 120 – 128, 2016.
17. F. Gavilan, R. Vazquez, and E. F. Camacho, "An iterative model predictive control algorithm for uav guidance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 3, pp. 2406–2419, 2015.
18. G. Sanchez, A. Limache, L. Giovanini, and M. Murillo, "Distributed model predictive control based on dynamic games," in *Advanced Model Predictive Control* (T. Zheng, ed.), ch. 4, pp. 65–90, InTech, 7 2011.
19. Joel Andersson, *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium, October 2013.

M. Murillo received the Ph.D degree in Engineering Science, computational intelligence, signals and systems mention from Universidad Nacional del Litoral (Argentina) in 2015. She is currently a post-doctoral student in the *Instituto de Investigacin en Señales, Sistemas e Inteligencia Computacional* (Research Institute for Signals, Systems and Computational Intelligence) at the Facultad de Ingeniería y Ciencias Hídricas, Universidad Nacional del Litoral (Argentina). Her research interests include nonlinear control theory and its applications to aerial vehicles simulation, navigation and control.

Guido Sánchez received the Ph.D degree in Engineering Science, computational intelligence, signals and systems mention from Universidad Nacional del Litoral (Argentina) in 2017. He is currently a post-doctoral student in the *Instituto de Investigacin en Señales, Sistemas e Inteligencia Computacional* (Research Institute for Signals, Systems and Computational Intelligence) at the Facultad de Ingeniería y Ciencias Hídricas, Universidad Nacional del Litoral (Argentina). His research interests include estimation and control of dynamical systems, multi-agent cooperative control and navigation and control of mobile robots.

Lucas Genzelis received the BSc degree in Informatics Engineering from Universidad Nacional del Litoral (Argentina) in 2015. He is currently a Ph.D. student at the *Instituto de Investigacin en Señales, Sistemas e Inteligencia Computacional* (Research Institute for Signals, Systems and Computational Intelligence) at the Facultad de Ingeniería y Ciencias Hídricas, Universidad Nacional del Litoral (Argentina). His research interests are mainly focused on autonomous navigation of flying vehicles and on identification of dynamical systems.

Leonardo Giovanini received the Ph.D. degree in Engineering Science from Universidad Nacional del Litoral (Argentina) in 2000. He was Senior Research Fellow and Technical Manager of the Industrial Control Center, University of Strathclyde (United Kingdom), from 2001 until 2007. In 2008 he joined to the *Instituto de Investigacin en Señales, Sistemas e Inteligencia Computacional* (Research Institute for Signals, Systems and Computational Intelligence), at the Universidad Nacional del Litoral, and the National Council of Technological and Scientific Research (CONICET). His research interests are mainly related to robust and nonlinear control theory including robust adaptive control, model predictive control and modelling and analysis of multi-agent systems. Application fields are robotics, autonomous vehicles and multi-agent systems.

path-following video .m4v

Click here to access/download

Video File

path_planning_husky_gazebo_video.m4v

path-following video .ogv

Click here to access/download
Video File
path_planning_husky_gazebo_video.ogv