A Path-Planning Algorithm Based on Receding Horizon Techniques

Marina Murillo
sinc(i), UNL-CONICET
Santa Fe, Argentina
mmurillo@sinc.unl.edu.ar

Guido Sánchez
sinc(i), UNL-CONICET
Santa Fe, Argentina
gsanchez@sinc.unl.edu.ar

Lucas Genzelis
sinc(i), UNL-CONICET
Santa Fe, Argentina
lgenzelis@sinc.unl.edu.ar

Leonardo Giovanini
sinc(i), UNL-CONICET
Santa Fe, Argentina
lgiovanini@sinc.unl.edu.ar

***Abstract*—In this article we present a 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 capabilities of the proposed path-planning algorithm, we explore the computation of a feasible path while following it with a Husky unmanned ground vehicle (UGV) using Gazebo simulator.**

***Index Terms*—Feasible optimal path, Model predictive control, Path-planning**

I. 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 issues of an autonomous vehicle. 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 [7] for an overview). Some recent path-planning algorithms can be found in [8]–[10]. In [8] *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 [9] 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 [10] *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 [11], *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 [12] *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 [13] 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.

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. In this article a unified framework to design a path-planning algorithm is presented. The proposed strategy can be summarized in Fig. 1. Using a simplified particle vehicle (PV) model, which is

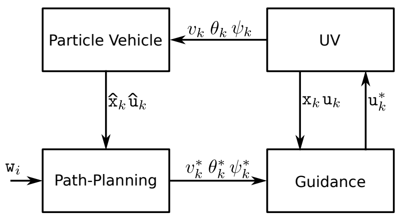

Particle Vehicle
$v_k\ \theta_k\ \psi_k$
UV
$\hat{\mathtt{x}}_k\,\hat{\mathtt{u}}_k$
$\mathtt{x}_k\,\mathtt{u}_k$
$\mathtt{u}_k^*$
$\mathtt{w}_i$
Path-Planning
$v_k^*\ \theta_k^*\ \psi_k^*$
Guidance

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

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 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 [14] 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 and iv) the inclusion of static obstacles into the path planning problem.

The organization of this article is as follows: in section II the 2D and 3D PV models are presented. In section III, the path-planning problem is introduced. In section IV a path-generation and path-following experimental example is outlined. Finally, in section V conclusions are presented.

II. Non-linear Particle Vehicle Model

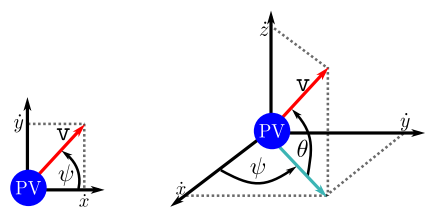
In this work we propose to use a PV model to obtain feasible and optimal paths for UVs. This section is devoted to obtain such a model for both the 2D and 3D cases. First, we provide a more general approach about systems representation and then we specify it for the case of 2D and 3D PV models.

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



(a) 2D (b) 3D

Fig. 2. Schemes of the proposed PV models

following state vector to model the 2D 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

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

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

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

[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 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 [10], [15], which can lead to unfeasible paths for a UV.

III. 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 [14] 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 [10], for example) 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 assumption that the control inputs of the PV have a limited rate of change, this figure shows how the path towards the waypoints[3] $\mathbf{w}_1$ and $\mathbf{w}_2$ is obtained. As shown in Fig. 3a, 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 $\mathbf{w}_2$. To obtain these sub-paths we propose to use the algorithm [14] to minimize the euclidean distance $(\mathrm{dist}(\mathbf{x}(t_j), \mathbf{w}_i))$ between the current position of the PV and the desired waypoint. Once $\mathbf{w}_1$ has been reached, 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. 3b and 3c). 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).

As it was mentioned before, we propose to modify the direction and modulus of the velocity vector of the PV using the control technique described in [14]. 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

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

where

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

$\mathbf{x}^r_{k|k}$, $\mathbf{u}^r_{k|k}$ 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). Subscripts $(\cdot)_{k+i|k}$ denote information at time $k+i$ which is predicted using the information available at time k. [5]. Receding horizon techniques use a cost function of the form

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

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:

$$\mathcal{L}_j(\mathbf{x}_{k+j|k}, \mathbf{u}_{k+j|k}) = \|\mathbf{x}_{k+j|k} - \mathbf{w}_i\|^2_{\mathbf{Q}_{k|k}} + \|\Delta\mathbf{u}_{k+j|k}\|^2_{\mathbf{R}_{k|k}} \tag{11}$$

and

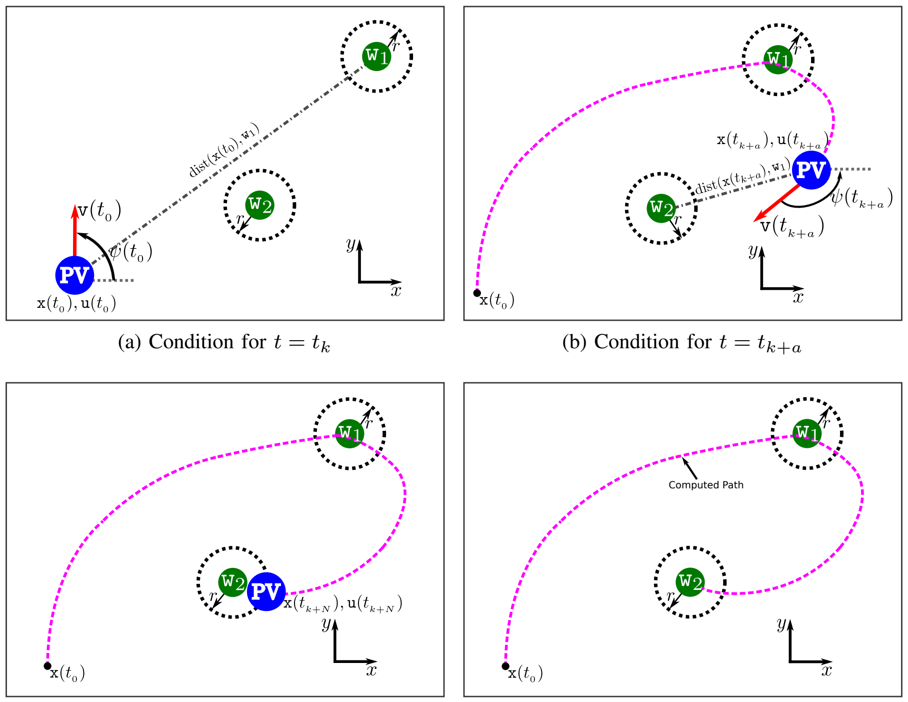
$$\mathcal{L}_N(\mathbf{x}_{k+N|k}) = \|\mathbf{x}_{k+N|k} - \mathbf{w}_i\|^2_{\mathbf{P}_{k|k}}, \tag{12}$$

[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.

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

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

[5]When it clearly refers to current time k, the time dependency at which the information is available will be omitted, i.e. $(\cdot)_{k+i|k} = (\cdot)_{k+i}$


W1
W2
PV
r
$\mathrm{v}(t_0)$
$\psi(t_0)$
$\mathrm{x}(t_0), \mathrm{u}(t_0)$
$\mathrm{dist}(\mathrm{x}(t_0), \mathrm{w}_1)$
$\mathrm{x}(t_{k+a}), \mathrm{u}(t_{k+a})$
$\mathrm{dist}(\mathrm{x}(t_{k+a}), \mathrm{w}_1)$
$\psi(t_{k+a})$
$\mathrm{v}(t_{k+a})$
$\mathrm{x}(t_{k+N}), \mathrm{u}(t_{k+N})$
$\mathrm{x}(t_0)$
Computed Path
y
x

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

(c) Condition for $t = t_N$ (d) Full computed path

Fig. 3. Computing a path between $\mathbf{x}(t_0)$ and $\mathbf{w}_2$ passing through $\mathbf{w}_1$

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. $\|(\cdot)\|^2_\alpha$ stands for the α-weighted 2-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 [14], we propose to solve the following optimization problem:

$$\min_{\tilde{\mathbf{U}}_k \in \mathcal{U}} \mathcal{J}(k)$$

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

where

$$\tilde{\mathbf{U}}_k{}^6 = [\tilde{\mathbf{u}}_{k|k}, \, \tilde{\mathbf{u}}_{k+1|k}, \, \cdots \, \tilde{\mathbf{u}}_{k+N-1|k}]^T \tag{14}$$

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. 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 [14]).

Remark 1. The convergence and stability issues for time-varying objective functions are guaranteed by the incorporation of the constraint $\mathcal{J}(k) \leq \mathcal{J}_0(k)$ [14].

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

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. 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 Fig. 4, the proposed path-planning algorithm is summarized.

IV. Experimental Results

In this section we explore the problem of computing a feasible path while following it with a Lego® Mindstorms® NXT[7] UGV configured as a skid steer vehicle as it is shown in Fig. 5.

Using the PV model (4) we solve the optimization problem (13) to find a 2D feasible and optimal path. To perform this example 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 $\mathbf{u}_0 = [0, \, 0]^T$. The PV model is discretized using a sampling rate $T_s = 0.1$ (s) and the horizon N was set to $N = 12$. The state and input weight matrices are chosen as $\mathbf{Q}_{k|k} = diag([10, \, 10, \, 10])$ and $\mathbf{R}_{k|k} = diag([0.1, \, 1.0])$, respectively. The PV constraints are configured as follows: $0 \leq \mathcal{T} \leq 2$ (N), $-5 \leq \Delta\psi \leq 5$ (deg/s),

[7]https://www.lego.com/en-us/mindstorms

```
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: Compute the optimal control input sequence $\tilde{\mathbf{U}}^{*,q}_k$ solving (13)
 3: Update $\mathbf{U}^{*,q}_{k|k} \leftarrow \mathbf{U}^{r,q}_{k|k} + \tilde{\mathbf{U}}^{*,q}_{k|k}$
 4: if $\left\|\mathbf{U}^{*,q}_k - \mathbf{U}^{*,q-1}_k\right\|_\infty \leq \epsilon$ then
 5:     $\mathbf{U}^*_k \leftarrow \mathbf{U}^{*,q}_k$
 6:     $k \leftarrow k + 1$
 7:     $q \leftarrow 0$
 8: else
 9:     $q \leftarrow q + 1$
10:     Update $\mathbf{U}^q_k = \mathbf{U}^{*,q-1}_k$
11:     Go back to line 2
12: end if
13: Apply $\mathbf{u}_{k|k} = \mathbf{u}^*_{k|k}$ to the system
14: Go back to line 1
```

Fig. 4. The Path-planning algorithm

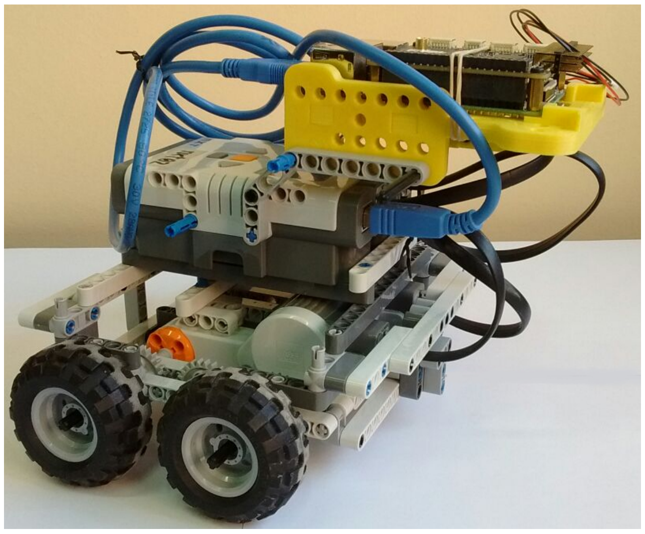


Fig. 5. Lego® Mindstorms ® NXT configuration

$-0.1 \leq \Delta\mathcal{T} \leq 0.1\,(\mathrm{N/s})$ and $0 \leq v \leq 1\,(\mathrm{m/s})$. 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 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 waypoints we set this area to a disk with a radius of $r = 0.05\,(m)$. We assume that the computed path should pass sufficiently close to the following waypoint[8]:

$$\mathbf{w}_1 = [0,\ 1.5,\ 0]^T. \tag{15}$$

We also consider the presence of a circular obstacle which is located at $\mathbf{c}_o = [0,\ 0.75]^T\,(\mathrm{m})$ with radius $r_o = 0.15\,(\mathrm{m})$. In order to take into account this obstacle in the path-planning stage we simply include the following constraint in problem (13):

$$(x_{k|k} - c_{o_x})^2 + (y_{k|k} - c_{o_y})^2 \geq r_o^2 \tag{16}$$

where c_{o_x} and c_{o_y} denote the x and y components of vector $\mathbf{c}_o$.

To solve the Lego® Mindstorms® NXT UGV guidance problem, we use the NMPC technique configured with a horizon $N = 5$ and the following mathematical model

$$\dot{\mathbf{x}} \;=\; f(\mathbf{x}, \mathbf{u}) \;=\; \begin{bmatrix} v_l \cos\psi_l \\ v_l \sin\psi_l \\ \omega_l \end{bmatrix}, \tag{17}$$

where $\mathbf{x} = [x_l, y_l, \psi_l]^T$ is the state vector, x_l and y_l denote the Lego® position and ψ_l denotes its yaw angle. The control input vector $\mathbf{u} = [v_l, \omega_l]^T$ includes the linear and angular velocities v_l and ω_l, respectively. It should be noticed that the model (17) used to control the Lego® Mindstorms® NXT robot is different from the plant, which is a much more complex model.

The estimation of the states was performed using only odometry and the Moving Horizon Estimation (MHE) technique presented in [16], with a horizon $N = 5$. Both, the NMPC and the MHE algorithms were executed in a Raspberry Pi 3 board[9]. The reference optimal path was computed offline and it was sent to the Raspberry Pi 3 using a TCP socket.

Assuming that both the PV and the Lego® UGV have the same initial position and orientation angle, the result of the Lego® path-following maneuver is shown in Fig. 6. As it can be seen, the Lego® UGV is able to follow the computed path quite well. This is mainly due to the fact that the path-planning algorithm generates a trajectory that is feasible and takes into account the dynamic constraints of the Lego® UGV. The errors appearing between the reference path and the Lego® position are due to the fact that we estimate states using odometry, i.e. we only use motor encoders (whose resolution is $1\,(\mathrm{deg})$) to estimate the states of the Lego® UGV, and because we do not consider the backlash of the gear train of the motors.

V. Conclusions

In this article we have presented a 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 kinematics, the dynamics

[8]As the experimental example is performed for the 2D case, the components of the following waypoint denote, respectively, *x*-coordinate, *y*-coordinate and speed.

[9]https://www.raspberrypi.org/

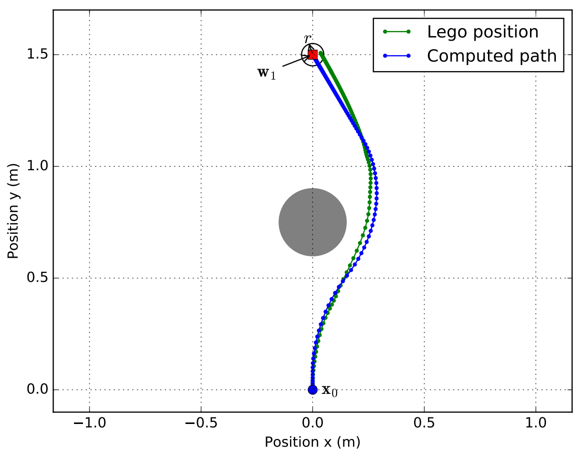


Fig. 6. Path-following with a Lego® Mindstorms® NXT robot

and constraints 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. Because we have used the iterated NMPC algorithm [14], optimality, stability and feasibility can be guaranteed. The performance and capabilities of the proposed path-planning algorithm were demonstrated through a path-generation and path-following experimental example. The path-following capabilities were explored using a Lego® Mindstorms® NXT UGV which is able to follow the computed feasible path. Considering that the estimation of the states of the Lego® was performed using only odometry, we consider that the experimental results obtained are satisfactory. As future work, we propose to add more sensors (like GPS and Inertial Measurement Units (IMUs)) to the system and to fuse data given by these sensors in order obtain more precision.

Acknowledgment

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, 2005, pp. 2–7. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1574593

[3] C. Zhang and J. Kovacs, “The application of small unmanned aerial systems for precision agriculture: a review,” 2012. [Online]. Available: http://link.springer.com/article/10.1007/s11119-012-9274-5

[4] S. M. Adams, M. L. Levitan, and C. J. Friedland, *High Resolution Imagery Collection Utilizing Unmanned Aerial Vehicles (UAVs) for Post-Disaster Studies*. American Society of Civil Engineers, 2013, ch. 67, pp. 777–793.

[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)*, April 2010, pp. 1–6.

[6] M. YAĞIMLI and H. S. Varol, “Mine detecting gps-based unmanned ground vehicle,” in *4th International Conference on Recent Advances in Space Technologies*, 2009, pp. 303–306.

[7] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[8] M. Saska, V. Spurný, and V. Vonásek, “Predictive control and stabilization of nonholonomic formations with integrated spline-path planning,” *Robotics and Autonomous Systems*, 2015.

[9] 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*, 2014, pp. 2287–2292.

[10] 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.

[11] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia, “Survey of robot 3d path planning algorithms,” *Journal of Control Science and Engineering*, vol. 2016, 2016.

[12] M. Hehn and R. D’Andrea, “Real-time trajectory generation for quadrocopters,” *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 877–892, 2015.

[13] V. T. Minh and J. Pumwa, “Feasible path planning for autonomous vehicles,” *Mathematical Problems in Engineering*, vol. 2014, 2014.

[14] M. Murillo, G. Sánchez, and L. Giovanini, “Iterated non-linear model predictive control based on tubes and contractive constraints,” *ISA Transactions*, vol. 62, pp. 120 – 128, 2016.

[15] 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.

[16] G. Sánchez, M. Murillo, and L. Giovanini, “Adaptive arrival cost update for improving moving horizon estimation performance,” *ISA transactions*, vol. 68, pp. 54–62, 2017.