J Intell Robot Syst (2012) 65:507–520
DOI 10.1007/s10846-011-9577-1

A Modular Framework for Fast Prototyping of Cooperative Unmanned Aerial Vehicle

Alessandro Benini · Adriano Mancini · Riccardo Minutolo · Sauro Longhi · Mauro Montanari

Received: 15 February 2011 / Accepted: 18 April 2011 / Published online: 18 August 2011

Abstract Unmanned Aerial Vehicles (UAVs) are gaining increasing interest thanks to their flexibility and versatility. However, these systems are very complex and a good simulation platform is needed. In this paper, a new Framework for simulation and fast prototyping of UAV control laws is presented. The Framework exploits the high realism of the simulations carried out in a three-dimensional virtual environment with the easiness of use of development systems such as Matlab® for fast prototyping of control systems. Then a novel method that exploits the benefits of Model Predictive Control (MPC) for cooperative scenarios is introduced. The obtained results show good performances of MPC in solving the formation problem of unmanned aerial vehicles; finding an optimal solution and taking into account different constraints. The developed framework allows also to easily change from simulated agent to real one.

Keywords UAV · MPC · Simulation · Fast prototyping

A. Benini (✉) · A. Mancini · S. Longhi
Dipartimento di Ingegneria dell'Informazione, Ancona, Italy
e-mail: a.benini@univpm.it

R. Minutolo · M. Montanari
Thales Italia S.p.A., Chieti, Italy

1 Introduction

Unmanned Aerial Vehicles (UAVs) are gaining an increasing interest in several important areas such as surveillance, rescue, replacing men in hazardous or difficult to reach environment. Moreover, in many cases, a cooperation of several robots to carry out more complex tasks such as location, photogrammetry is required. Given the complexity of control laws that regulate the dynamics of these aircraft and the difficulty of designing robust control systems, a very good simulation platform is necessary. Simulation activities are essential by making it possible to study and compare different approaches in solving a particular problem, providing not only a drastic reduction in development time but also in costs. Matlab/Simulink is an excellent choice for rapid prototyping of control systems. However, in the Simulink environment the mathematical model of the system under simulation is required. These mathematical models are in some cases too difficult to manage or not available at all. Moreover, it is difficult to model mathematically complex robot behaviours in realistic environments, as in the case of cooperative tasks. For such situations a large number of three-dimensional simulators has been developed:

- Player/Stage Gazebo [1]: the Player Project based on ODE physic engine (formerly the Player/Stage Project or Player/Stage/Gazebo

Project) is a project to create free software for research into robotics and sensor systems. Its components include the Player network server and Stage/Gazebo robot platform simulators.

- FlightGear [2]: FlightGear Flight Simulator (often shortened in FlightGear or FGFS) is a free, open-source and multi-platform flight simulator developed by the FlightGear project since 1997. The simulation engine in FlightGear is called SimGear. It is used both as an end-user application and in academic and research environments. Flight Dynamics Models (FDM) are how the flight of an aircraft is simulated in the program. FlightGear uses a variety of internally written and imported flight model projects. Any aircraft must be programmed to use one of these models.
- Microsoft Robotics Developer Studio [3]: the Microsoft® Robotics Developer Studio 2008 R3 (Microsoft RDS) is a Windows®-based environment for academic, hobbyist, and commercial developers to easily create robotics applications across a wide variety of hardware. The integration of the NVidia PhysX Technologies enables leveraging a very strong physics simulation.
- SimplySim SimplyCube [4]: the software integrates nicely with the Microsoft Robotics Developer Studio providing a complete and detailed 3D environments and sensor models. In fact, any 3D environment created using one of the many tools included in Simply Suite can be exported and used in a robotic simulation within MRDS. The SimplySim SimplyCube is currently implemented for two of the most accurate physic engines on the market: NVidia PhysX [5] and Newton Game Dynamics [6].
- X-Plane: X-Plane [7] is a flight simulator produced by Laminar Research for Android, IOS, Linux, Mac OS X and Windows. X-Plane can be bundled with other software to create and edit new aircraft and scenery. The flight model of X-Plane is based on the so-called "Blade Element Theory": In practice, the aircraft aerodynamic surfaces (wings, tail surfaces, etc.) are split into several parts, then the force acting on each one of them is calculated by applying a fluid.

This work extends the previous framework [8, 9] that was based on modularity and stratification in different specialized layers; the limitation was the missing of a physical engine and an advanced 3D visualization interface.

Among all the solutions outlined above, the SimplySim provides good opportunities and it can be programmed using the language of the framework .NET. In this way, it is possible to interface it with Microsoft Robotics Developer Studio and other platforms. Further it includes a full suite of tools for the realization not only of three-dimensional models of robots (also in terms of physical characteristics) but also for the realization of complete three-dimensional scenes. This makes it quite interesting for the creation of highly realistic simulation systems. In this paper, a modular framework, based on SimplySim© SimplyCube for fast prototyping of cooperative Unmanned Aerial Vehicle is presented. The developed framework combines the high realism of the simulations carried out in a three-dimensional virtual environment (in which the most important laws of physics act) with the easiness of Simulink for fast prototyping of control systems. This paper is organized in the following way. In the next section a brief overview of developed framework is provided. Section 3 deals with Simulator interfacing with Matlab. Section 4 presents Networked Decentralized Model Predictive Control for formation control. In Section 5 the developed Framework is used for testing ND-MPC and, in the last one, conclusions and future avenues for further research in this area are explored.

2 The Framework

The developed framework provides a set of features mainly oriented to the simulation and control of autonomous aircraft in cooperative tasks. As mentioned before the SimplyCube environment offers the opportunity to develop own applications exploiting the power of language based on .NET platform (C#, Visual Basic and C++/CLI). Thanks to its easiness of use and the considerable support provided for it, the C# is the ideal candidate for the development of applications based on SimplyCube. The framework consists of a series of

modules, each of which is specialized in a specific task. In the current version of the framework the modules available are:

- *Management of three-dimensional environment actors (both static and dynamic) Library*: This module implements classes that allow 3D models to interact with the virtual world. The distinction between static and dynamic actors allows to divide entities into two separate categories: items on which it is possible to apply forces and items for which this is not possible.
- *Drone Library*: This library provides a set of capabilities for the management of quadrotors. The classes implemented allow to obtain information about all aspects of the quadrotor (for example yaw, pitch and roll angles) as well as methods for handling them. At each 3D model an XML configuration file is associated. In this file it is possible to edit any physical parameter of the aircraft (viscous friction of air on the wings, maximum rpm for each engine, and so on).
- *Avionic Instruments Library*: This module plays a minor role. It allows graphical display of data for each quadrotor. In this way we can gain real-time information concerning, for example, the position in space of an aircraft. In the current version of the framework, the following two tools are provided: artificial horizon and altimeter. Thanks to the modularity of the software, we can add a new instrument at any time simply by developing the code that will inherit the basic methods from the parent class.
- *Network Services Library*: this library provides methods for creating TCP/IP and UDP/IP connections.

Together with these modules, within the framework there is a specific library to interface the simulator with Matlab/Simulink. This open-source library, called PNET [10], can be used to set up TCP/IP and UDP/IP connections with Matlab. It can transmit data over the Intranet/Internet between Matlab processes or other applications.

The following figure shows the structure of the framework just discussed (Fig. 1).

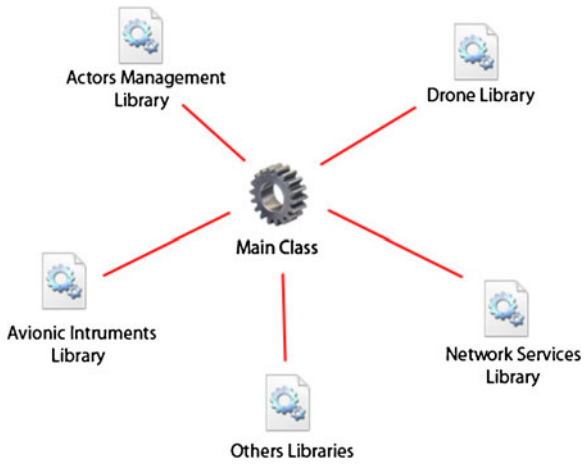


Fig. 1 Framework structure diagram

2.1 Management of Three-Dimensional Environment Actors Library

As mentioned before this module implements the classes that allow 3D models to interact with the virtual world. A physical object is a rigid body that can collide with other objects. In the SimplyCube environment they are called actors and mainly separated into two categories:

- Static Actors;
- Dynamic Actors.

The dynamic actors can move. Their properties are mass, velocity and inertia. Moreover, it is possible to apply a force or a torque to them. When a collision is detected in a dynamic actor, the simulator applies a force and a torque to it in order to simulate a real reaction, keeping in consideration the properties of mass and inertia. The forces and torques applied dynamically change the speed of the actor and therefore the position. The static actors are much simpler than dynamic actors. In fact, for them reaction forces and torques are not calculated and they do not have the opportunity to travel in space.

As mentioned, a rigid body can collide with other objects. To make this happen, it is necessary to define an area of collision, called "collision shape". SimplyCube environment provides three different collision shapes: parallelepiped, ellipsoid and capsule. By combining these three basic areas it is possible to generate arbitrarily complex

surfaces collision. These areas are included inside the actors (both dynamic and static).

2.2 Drone Library

This library provides a set of capabilities for the management of quadrotors. For each 3D model of quadrotors, an XML configuration file is associated. In order to create a drone, it is necessary to declare two files:

- A file that defines the complex object which will make the drone: It is an assembly of several simple actors linked with joints;
- A file which will define the drone configuration: its body (main part), its rotors, and for each rotor the engine and blade configuration.

A drone is a complex object composed of a body, several rotors and blades. In order to make the drone fly, a force is applied on each rotor depending on its angular velocity (rotation speed). For each rotor the force applied equals to: $F = w^2 k$ where:

- F is the force applied, in Newton's;
- w is the angular velocity of the blade, in $\mathrm{rad/s}^{-2}$
- k is a coefficient, calculated as follow: $k = \frac{MassLift \cdot Gravity}{RPMLift^2}$

Varying *MassLift* and *RPMLift*, it is possible to define a different k coefficient for each rotor. The *RPMLift* can be set arbitrarily: it only represents the RPM reference for a rotor, but *MassLift* should be well estimated in order to make the drone stable.

2.3 Avionic Instruments Library

This library contains all the classes used for the development and maintenance of avionic instruments. The tools developed include:

- An artificial horizon;
- An altimeter.

A set of primitives for the creation of other types of instruments are provided, also to extend—in future development—the avionic library. All virtual instruments have been made following the same logic: objects are made with a series of bitmap images which are rotated, translated or scaled before being displayed. The methods for the operations of rotation, translation and scaling are implemented in a parent class. Subsequently, each class for different virtual instruments inherits from the base class and uses its parameters to manipulate the images. Specifically the rotation of the image is divided into two parts:

- Effective Rotation around a fixed point which is user-defined;
- Translation of the image at a given point (which is always defined by the user).

2.4 Network Services Library

The library implements the communication layer based on TCP/IP and UDP/IP. The advantages of using these protocols for communication between the framework and the outside world are various; they allow for the possibility to create a communication protocol shared by all parts of the system and also allow for the option to sort the computational load on several computers which do not have to be located in the same network.

3 Simulator Interfacing with Matlab

In this chapter the methodology used in the design of process control systems is described. After a brief discussion about the communication between simulator and control system, an overview of PID control system for attitude control and tracking position of a quadrotor is discussed.

3.1 The Quadrotor Model

The context of this work is the assumption of a gray-box modelling. The process is a 3D model of the aircraft within the simulation environment. The system of equations describing the dynamics of the aircraft was replaced with an S-function (Fig. 2) whose input and outputs variables are reported in Table 1. The S-function implements a UDP server to exchange data with the simulator; the quadrotor is considered as a gray-box due to the limited availability of parameters.

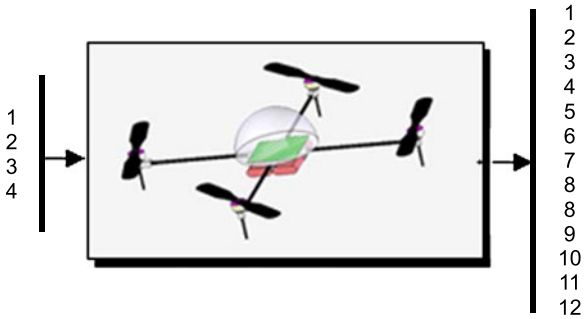


Fig. 2 Simulink block of quadrotor

The logic behind the S-function is as follows:

(1) The simulator sends sensor data to the server UDP contained within the S-function;
(2) The S-function transmits data received from the simulator out of the Simulink block and provides values for The calculation of control laws;
(3) The control system calculates the control laws and provide the results in input to the S-function;
(4) The UDP server included in the S-function sends back to the simulator the values of control law;
(5) The simulator uses the data received to set the value of rotation of the rotors.

Table 1 Input and Outputs list of S-function

	Description
Input	
1	Δrpm for altitude
2	Δrpm for pitch angle
3	Δrpm for roll angle
4	Δrpm for yaw angle
Output	
1	Position among x
2	Position among y
3	Position among z
4	Linear speed among x
5	Linear speed among y
6	Linear speed among z
7	Pitch angle
8	Roll angle
9	Yaw angle
10	Angular speed around pitch
11	Angular speed around roll
12	Angular speed around yaw

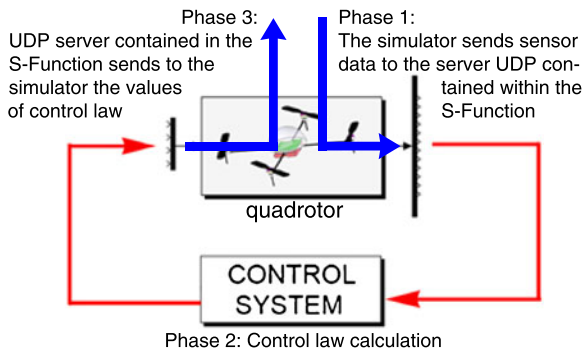


Fig. 3 Data flow between simulator and Simulink control system

Figure 3 shows the data flow between the control system and the simulation system.

3.2 A PID Controller for Position Tracking

For the position of the quadrotor, a control system consisting of three nested control loops was realized. The inner ring is responsible for the stabilization of the aircraft with the values of pitch, roll and yaw angles. The reference for the attitude control is provided by the speed controller, which takes as a reference the efforts of control position. Figure 4 shows the control architecture of the control system for stabilization and position tracking of a quadrotor.

Controllers for both, position and speed, are done by PID algorithms. The calibration of parameters was made with heuristic techniques, just as for attitude control. Using these three nested control loops, it is possible to move the aircraft along the way and let it to reach the waypoints with zero speed. For the generation of references of the position control, the Simulink Signal Builder was used. This tool provides the ability to define separately the coordinate values of positions to reach for the aircraft. In this way it is possible to operate with absolute simplicity the generation of trajectories for the quadrotor.

4 Formation Control via Networked Decentralized Model Predictive Control

The structure of the proposed framework allows the development of cooperative control laws.

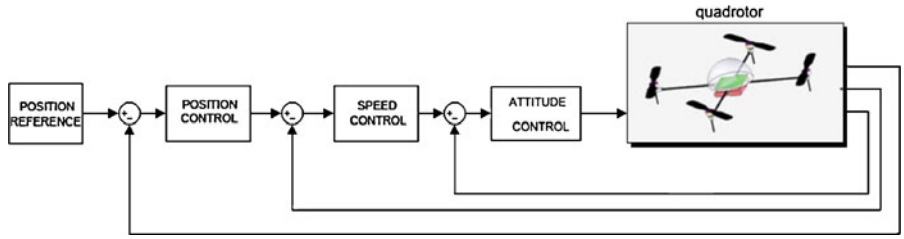


Fig. 4 Control system for stabilization and position tracking of quadrotor

These control techniques include PID and Model Predictive Control. In this section, an algorithm for MPC formation flight of two quadrotor is presented. Formation Control has been addressed in [11–20]. However, in the near future it will be possible to develop control techniques using formation control not only for UAVs but also for UGVs.

In Control Engineering, when the process to be monitored is large or too fast to implement via classical system of centralized control, the problem of control is often divided into several sub problems where the requirements of the specification is guaranteed by a proper collaboration of control subsystems. This control technique called Decentralized Control has been widely studied in recent years, mainly because of the remarkable expansion of computer networks. The coordination of the control subsystems can be obtained through the exchange of information among agents inside a communication network. An example of a formation fleet is shown in Fig. 5 where V^j is the leader for the quadrotor V^i. In this way the coordination is completely

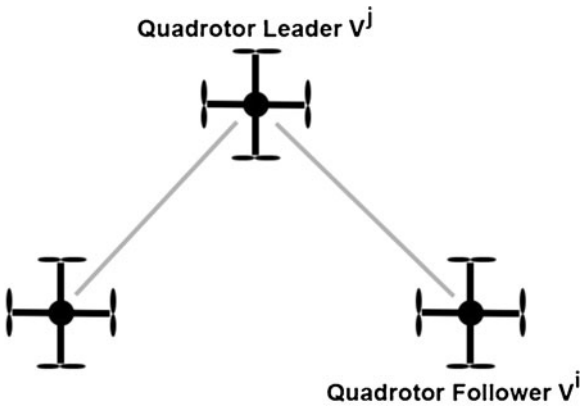


Fig. 5 A formation of quadrotor in a leader-follower scheme

decentralized and also the control strategy. The features of Networked Decentralized Model Predictive Control (ND-MPC) have recently been presented and successfully tested in several real cases in which it was necessary to have a strong interaction among a large number of subsystems. The advantages of using the MPC are linked to the ability to generate control actions taking into account not only the information from each subsystem but also the non-linearity and any limitations of the system. In this section the ND-MPC is used to solve the problem of leader-follower of two quadrotors trying to minimize the control actions for each of the components. The basic-idea is the following: vehicles must stay at a constant distance from each other: each vehicle follows the leader and only the leader knows the path. Each aircraft implements a Decentralized MPC algorithm based on the information collected by its sensors and information from other aircraft on the network. An error model is defined to ensure that the quadrotors remain in formation [21, 22]. These models are highly non linear, coupled and dynamically bound. The solution to the considered problem can be formulated in a general way for a set of quadrotors described in the following.

4.1 The Kinematic Model

Let's consider a set of N aircraft whose configuration at the generic instant t is denoted by the following vector:

$$\mathbf{q}^i(t) \triangleq \left[q_x^i(t)\ q_y^i(t)\ q_\theta^i(t)\right]^T \tag{1}$$

Assuming that a low-level controller for speed maintaining is defined, the control problem is transformed into a desired route planning for low-level controller which should define the optimal value of the linear speed v and rotational speed ω

and should ensure that the aircraft remain in formation minimizing the efforts as much as possible. Therefore, each aircraft will be guided through its linear velocity and angular velocity that will define the vector of control actions:

$$\mathbf{u}^i(t) = \left[v^i(t)\ \omega^i(t)\right]^T \tag{2}$$

The kinematics of each single aicraft is described by the following continuous-time model:

$$\dot{\mathbf{q}}^i(t) = \begin{bmatrix} \cos q^i_\theta(t) & 0 \\ \sin q^i_\theta(t) & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}^i(t). \tag{3}$$

By sampling Eq. 3 with a sample time T_s, velocities v, w produce finite linear and angular displacement $v^i_k \triangleq v^i(kT_s)T_s$, $w^i_k \triangleq w^i(kT_s)$ T_s, within each sampling interval. Defining $\mathbf{q}^i_k \triangleq [q^i_{x.k}\ q^i_{y,k}\ q^i_{\theta,k}]^T \triangleq \mathbf{q}^i(kT_s)$, $\mathbf{u}^i_k \triangleq [v^i_k\ w^i_k]^t \triangleq \mathbf{u}^i(kT_s)T_s$ and approximating the derivatives with a proper discretization methods, the following discrete-time model is obtained:

$$\mathbf{q}^i_{k+1} = \mathbf{q}^i_k + \mathbf{H}^i_k \mathbf{u}^i_k \tag{4}$$

where, in general,

$$\mathbf{H}^i_k = \mathbf{H}(q^i_k) = \begin{bmatrix} \cos q^i_{\theta,k} & 0 \\ \sin q^i_{\theta,k} & 0 \\ 0 & 1 \end{bmatrix} \tag{5}$$

When dealing with formation control problem, the position of a leader aircraft with respect to the follower aircraft should be kept equal to a desired value. Let's define a rotation matrix operator which transforms the fixed frame coordinates into rotated frame coordinates by a rotation α as follows:

$$\mathbf{T}(\alpha) \triangleq \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

With respect to Fig. 6, denoting with $\mathbf{T}^i_k \triangleq \mathbf{T}(q^i_{\theta,k})$ the rotational matrix which changes inertial coordinates into the frame reference $(\mathbf{O}^i, \mathbf{x}^i, \mathbf{y}^i)$ fixed to aircraft V^i, the relative displacement of aircraft V^j referred to V^i is:

$$\mathbf{d}^{ij}_k \triangleq \left[x^{ij}_k\ y^{ij}_k\ \theta^{ij}_k\right]^T = \mathbf{T}^i_k \left(\mathbf{q}^j_k - \mathbf{q}^i_k\right), \tag{7}$$

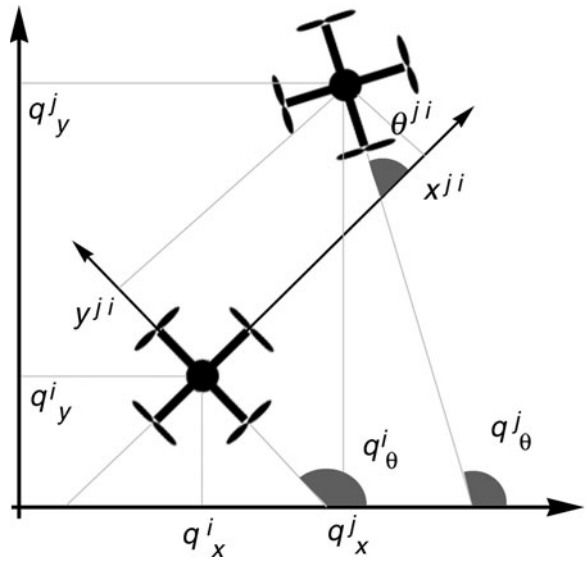


Fig. 6 Relative configuration of aircraft V^j with respect to aircraft V^i for the tracking error system

which implies by Eq. 4 the following relation:

$$\mathbf{d}^{ij}_{k+1} = \mathbf{T}^i_{k+1}\left[\mathbf{q}^j_k + \mathbf{H}^j_k \mathbf{u}^j_k - \mathbf{q}^i_k - \mathbf{H}^i_k \mathbf{u}^i_k\right]. \tag{8}$$

Defining

$$\mathbf{A}^i_k \triangleq \mathbf{A}\left(\mathbf{u}^i_k\right) \triangleq \mathbf{T}^i_{k+1}\left(\mathbf{T}^i_k\right)^{-1} \tag{9}$$

$$\mathbf{B}^i_k \triangleq \mathbf{B}\left(\mathbf{u}^i_k\right) \triangleq -\mathbf{T}^i_{k+1}\mathbf{H}^i_k \tag{10}$$

$$\mathbf{E}^{ji}_k \triangleq \mathbf{E}\left(\mathbf{d}^{ji}_k, \mathbf{u}^i_k, \mathbf{u}^j_k\right) \triangleq \mathbf{T}^i_{k+1}\mathbf{H}^j_k, \tag{11}$$

Eq. 8 for $i, j = 1, \ldots, N,\ j \neq i$ gives the formation vector model:

$$\mathbf{d}^{ji}_{k+1} = \mathbf{A}^i_k \mathbf{d}^{ji}_k + \mathbf{B}^i_k \mathbf{u}^i_k + \mathbf{E}^{ji}_k \mathbf{u}^j_k. \tag{12}$$

Note that matrices $\mathbf{A}^i_k$, $\mathbf{B}^i_k$, $\mathbf{E}^{ji}_k$ are, in general, functions of the current displacement $\mathbf{d}^{ji}_k$, control action $\mathbf{u}^i_k$ and interaction vector $\mathbf{u}^j_k$. In this case, because there is not a relative rotation between quadrotors, the previous Eq. 12 is simplified as follows:

$$\mathbf{d}^{ji}_{k+1} = \mathbf{d}^{ji}_k + \mathbf{u}^i_k + \mathbf{u}^j_k. \tag{13}$$

4.2 The Predictive Model

The h-head state prediction for aircraft V^i is computed by h iterations of model (12) which gives:

$$\widehat{\mathbf{d}}^{ji}_{k+1|k} = \widehat{\mathbf{A}}^{i}_{k|k}\widehat{\mathbf{d}}^{ji}_{k|k} + \widehat{\mathbf{B}}^{i}_{k|k}\widehat{\mathbf{u}}^{i}_{k|k} + \widehat{\mathbf{E}}^{ji}_{k|k}\widehat{\mathbf{u}}^{j}_{k|k}, \tag{14}$$

$$\begin{aligned}\widehat{\mathbf{d}}^{ji}_{k+h|k} &= \widehat{\mathbf{B}}^{i}_{k+h-1|k}\widehat{\mathbf{u}}^{i}_{k+h-1|k} + \widehat{\mathbf{E}}^{ji}_{k+h-1|k}\widehat{\mathbf{u}}^{j}_{k+h-1|k} \\ &\quad + \sum_{l=1}^{h-1}\prod_{n=1}^{h-l}\widehat{\mathbf{A}}^{i}_{k+h-n|k} \\ &\quad \times \left[\widehat{\mathbf{B}}^{i}_{k+l-1|k}\widehat{\mathbf{u}}^{i}_{k+l-1|k} + \widehat{\mathbf{E}}^{ji}_{k+l-1|k}\widehat{\mathbf{u}}^{j}_{k+l-1|k}\right] \\ &\quad + \prod_{n=1}^{h}\widehat{\mathbf{A}}^{i}_{k+h-n|k}\widehat{\mathbf{d}}^{ji}_{k|k}\end{aligned} \tag{15}$$

4.3 Physical Constraints

Due to physical limits, the discrete time-model (12) is subject to a set of constraints on its velocities and accelerations. For each integer $h \geq 1$, the discrete angular and linear velocities are constrained by the following inequalities:

$$\underline{v}^i \leq v^i_{k+h-1} \leq \overline{v}^i, \qquad \underline{w}^i \leq w^i_{k+h-1}, \leq \overline{w}^i \tag{16}$$

$$\left|\Delta v^i_{k+h-1}\right| \leq \overline{\Delta v}^i, \qquad \left|\Delta w^i_{k+h-1}\right| \leq \overline{\Delta w}^i \tag{17}$$

where Δ is a difference operator such that $\Delta v^i_{k+h-1} \triangleq v^i_{k+h-1} - v^i_{k+h-2}$ and $\Delta w^i_{k+h-1} \triangleq w^i_{k+h-1} - w^i_{k+h-2}$ gives the discrete accelerations.

4.4 The Leader-Follower Problem

Let us assume that the aircraft V^i is controlled by a local independent controller A^i which implements an MPC strategy. The reference formation pattern is defined by vectors

$$\overline{\mathbf{d}}^{ji} \triangleq \left[\overline{x}^{ji}\ \overline{y}^{ji}\overline{\theta}^{ji}\right]^T, \tag{18}$$

which specifies the desired displacement for the couple of aircraft V^i, V^j, where V^j is the leader for $V^i, i = 0, \ldots, N$, $j \neq i$. In order to keep the desired formation, agent A^i communicates with the other agents and iteratively computes the optimal control sequence $\widehat{\mathbf{u}}^{i}_{\cdot|k} \triangleq [(\widehat{\mathbf{u}}^{i}_{k|k})^T\ \ldots\ (\widehat{\mathbf{u}}^{i}_{k+p-1|k})^T]^T$ over the horizon p.

The following framework is proposed here:

- Each control agent $A^i, i = 1, \ldots, N$ communicates with its neighboring agents;
- Each control agent $A^i, i = 1, \ldots, N$ knows its configuration $\mathbf{q}^i$ and the configurations $\mathbf{q}^j$ of the neighboring agents.
- The reference trajectory T^*, to be followed by the main leader aircraft V^1, is generated by a virtual reference aircraft V^0 which moves according to the considered model (4).
- each V^i, $i = 2, \ldots, N$ follows one and only one leader V^j, $j \neq i$; V^1 follows virtual vehicle V^0 which exactly tracks the reference trajectory T^*.
- each V^i, $i = 1, \ldots, N$ should keep the formation vector $\overline{d}^{ji}$ from its leader V^j.

In order to evaluate the performance of a follower A^i, a measure of difference between the actual or predicted formation vector $\mathbf{d}^{ji}_k$ and the constant reference vector $\overline{d}^{ji}$ is needed. Given the actual formation vector $\mathbf{d}^{ji}_k \triangleq [x^{ji}_k\ y^{ji}_k\ \theta^{ji}_k]^T$ of vehicle V^i which follows its leader V^j with the desired formation vector $\overline{\mathbf{d}}^{ji}_k \triangleq [x^{ji}\ y^{ji}\ \theta^{ji}]^T$ the following scalar is chosen as a measure of the performance for control Agent A^i:

$$\begin{aligned}\left\langle \mathbf{d}^{ji}_k - \overline{\mathbf{d}}^{ji} \right\rangle^2 &\triangleq \rho_x \left(x^{ji}_k - \overline{x}^{ji}\right)^2 + \rho_y \left(y^{ji}_k - \overline{y}^{ji}\right)^2 \\ &\quad + \rho_\theta \sin^2 \frac{\theta^{ji}_k - \overline{\theta}^{ji}}{2}\end{aligned} \tag{19}$$

4.5 Decentralized MPC

Given the tree of connections $\mathbf{g} = [g^1, \ldots, g^N]$ of aircraft, where g^i denotes the leader of the quadrotor $i, i = 1, \ldots, N$, reference trajectory T^* and prediction horizon p, the Networked Decentralized MPC problem at time k for the set of aircraft $\{V^1, \ldots, V^N\}$ with weighting coefficient $\mu^i \in \mathbb{R}, i = 1, \ldots, N$ consists in solving N independent non linear optimization problems stated, for $i = 1, \ldots, N$, as:

$$\min_{\widehat{\mathbf{u}}^{i}_{\cdot|k}} J^i_k \left(\mathbf{d}^{ji}_k, \widehat{\mathbf{u}}^{i}_{\cdot|k}, \mathbf{u}^{j*}_{\cdot|k-1}\right), \tag{20}$$

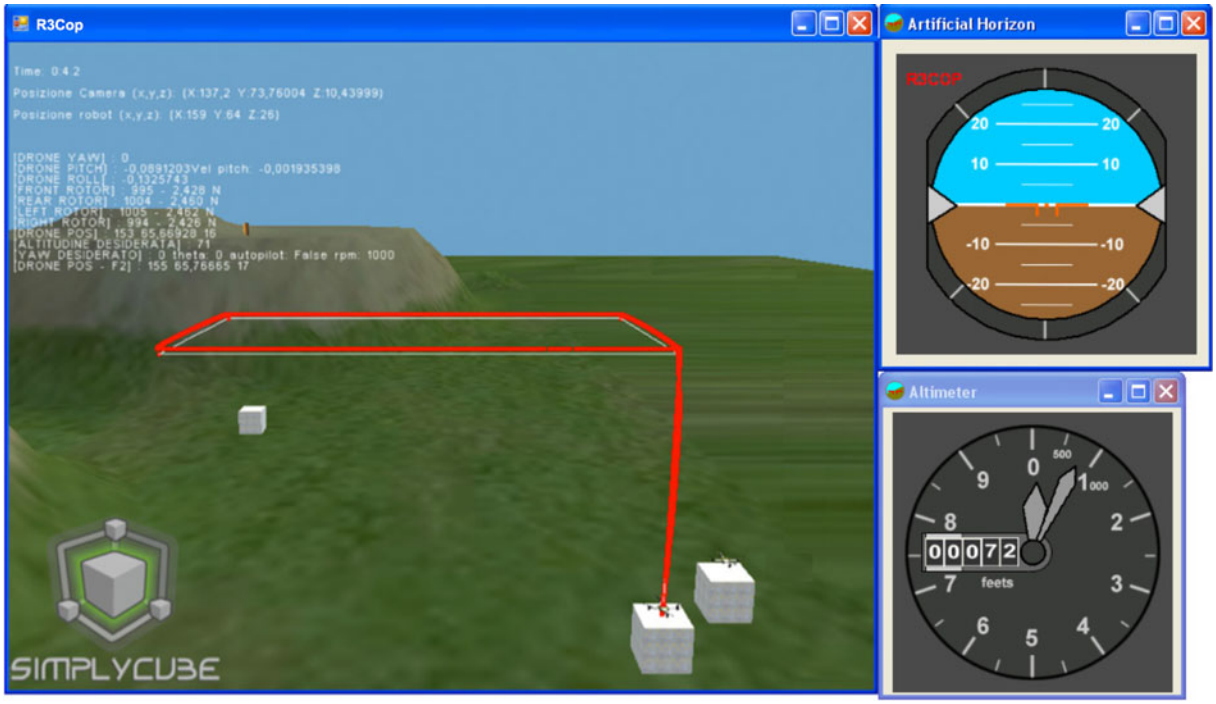


Fig. 7 A sample of scenario where two UAV have performed a path following by MPC control law

where

$$J_k^i\left(\mathbf{d}_k^{j\bar{i}}, \widehat{\mathbf{u}}_{\cdot|k}^i, \mathbf{u}_{\cdot|k-1}^{j*}\right) \triangleq \sum_{h=1}^{p}\left\langle \widehat{\mathbf{d}}_{k+h|k}^{g^i,i} - \overline{\mathbf{d}}^{g^i,i}\right\rangle^2 + \mu^i \left|\widehat{\mathbf{u}}_{k+h-1|k}^i\right|^2, \tag{21}$$

subject to:

$$\underline{v}^i \le v_{k+h-1}^i \le \overline{v}^i, \qquad \underline{w}^i \le w_{k+h-1}^i, \le \overline{w}^i \tag{22}$$

$$\left|\Delta v_{k+h-1}^i\right| \le \overline{\Delta v}^i, \qquad \left|\Delta w_{k+h-1}^i\right| \le \overline{\Delta w}^i \tag{23}$$

Control actions thus determined will act directly on the control of pitch and roll angles of each of the N aircraft. A simulation scenario where the absolute bound for speed among x and z axis has been set to 3 m/s is shown in Fig. 7.

5 Main Results

In this section the results about formation control of two aircraft are presented. The leader aircraft will follow a virtual trajectory generated by a virtual reference aircraft V^0. In Fig. 8 the considered path in the simulation is shown.

Simulations last 180 s during which leader aircraft takes off, flies the path and lands at the point of departure. The simulation time is divided into ten intervals T_i, $i = 1, ..., 10$ described as follow:

- $T1 = [0–20]$ s: the leader takes off and reach the waypoint 1 (WP1)
- $T2 = [20–30]$ s: the leader remains in WP1
- $T3 = [30–50]$ s: the leader flies the distance between WP1 and WP2
- $T4 = [50–70]$ s: the leader remains in WP2
- $T5 = [70–80]$ s: the leader flies the distance between WP2 and WP3
- $T6 = [80–100]$ s: the leader remains in WP3
- $T7 = [100–120]$ s: the leader flies the distance between WP3 and WP4
- $T8 = [120–135]$ s: the leader remains in WP4
- $T9 = [135–145]$ s: the leader flies the distance between WP4 and WP1
- $T10 = [160–175]$ s: the leader lands.

In the following the results of ND-MPC parameters tuning are shown. With the framework developed, it is possible to make all the simulations needed to calculate the optimal values of the

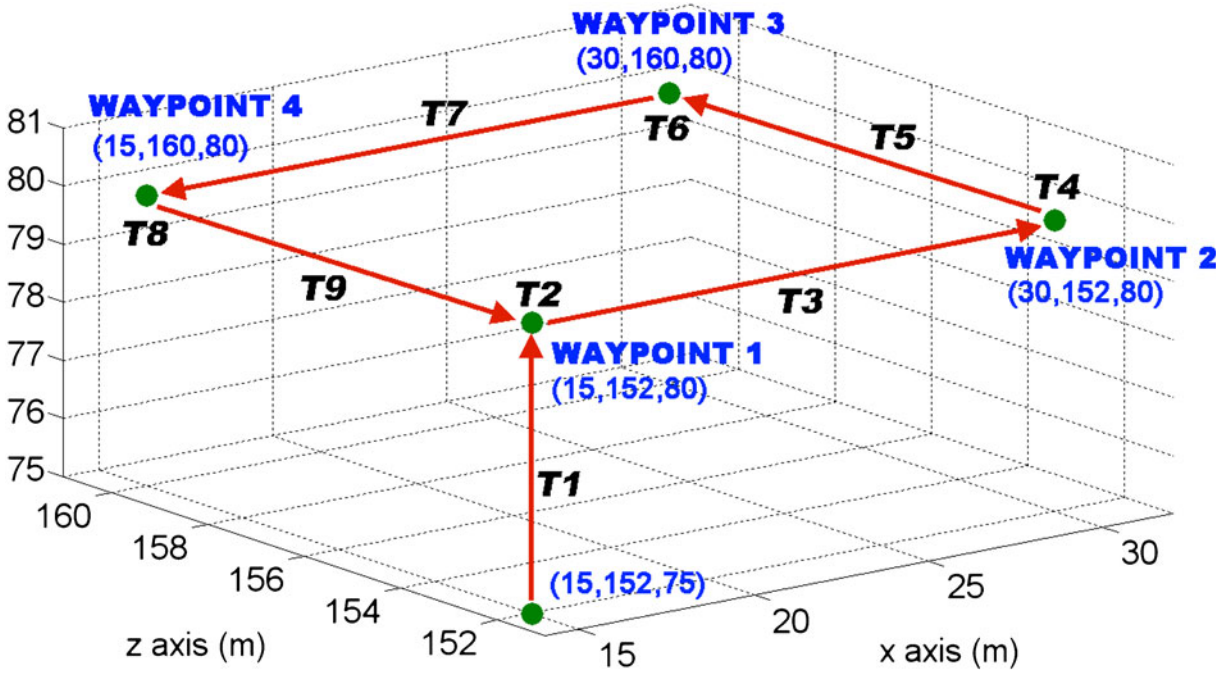


Fig. 8 Path of the virtual aircraft V^0 that is leader for aircraft V^1

parameters for the functional J contained in the ND-MPC algorithm. About the simulations the trend of relative positioning error among the x and z axis (pitch and roll respectively) will be shown.

5.1 Simulation Results

In these simulations the leader aircraft follows the rectangular path defined in the previous paragraph. The follower aircraft is constrained to maintain a relative distance from the leader of 2 m along the x axis and 2 m along the z axis. Tables 2 and 3 show the positioning relative errors of the follower respect to the leader: for 2 different choices of horizon prediction. In simulation #15, the values of the chosen parametrs ensure a maximum error of 0.12 m among the two coordinates with an horizon prediction of 20. The values of RMS_X and RMS_Z are referred to the entire flight sequence. Values of $\max_{ErrX}$ and $\max_{ErrZ}$ are expressed in meters (m).

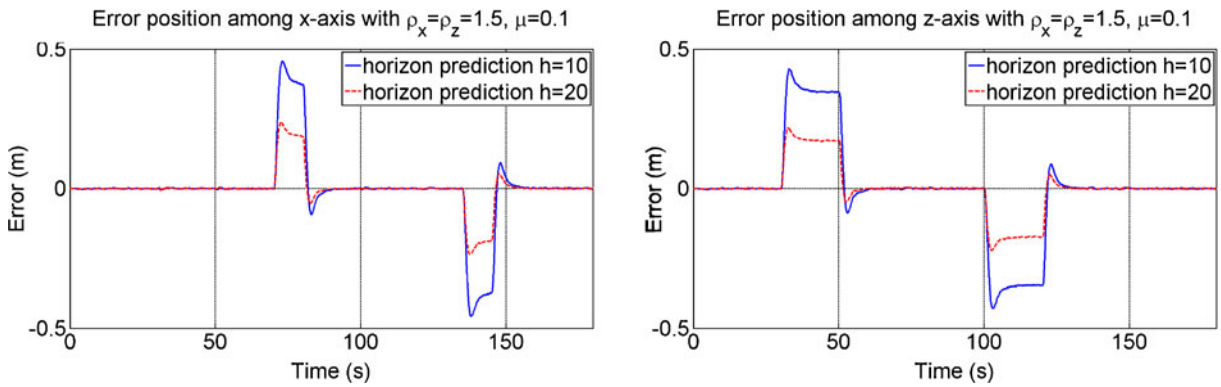
Figures 9, 10 and 11 show the relative position error among x and z axis. Each graph represents the evolution of two simulations carried out using the same weight coefficients for relative positioning errors along the two axes and two different values for the prediction horizon.

Table 2 Simulation results of MPC with horizon prediction equals to 10

Simulation number	ρ_x ρ_z	μ	h	$\max_{Errx}(m)$	RMS_X	$\max_{Errz}(m)$	RMS_Z
1	0.5	0.1	10	1.19	0.36	1.11	0.48
2	0.5	0.5	10	4.35	1.29	4.85	2.01
3	1.5	0.1	10	0.45	0.12	0.42	0.16
4	1.5	0.5	10	1.85	0.56	1.71	0.77
5	3.5	0.1	10	0.21	0.05	0.19	0.07
6	3.5	0.5	10	0.89	0.26	0.82	0.34
7	3.8	0.1	10	0.19	0.05	0.18	0.06
8	3.8	0.5	10	0.83	0.24	0.77	0.32

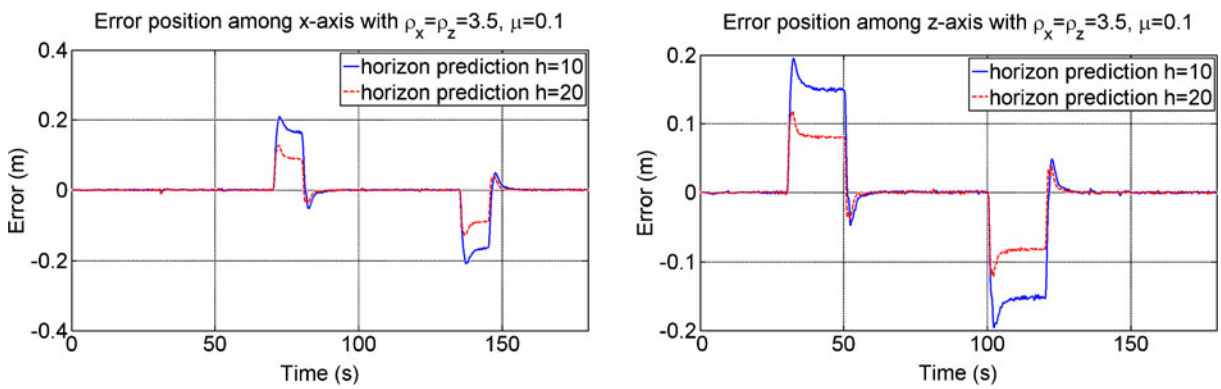
Table 3 Simulation results of MPC with horizon prediction equals to 20

Simulation number	ρ_x ρ_z	μ	h	$\max_{Errx}(m)$	RMS_X	$\max_{Errz}(m)$	RMS_Z
9	0.5	0.1	20	0.61	0.18	0.57	0.23
10	0.5	0.5	20	2.52	0.74	2.38	1.05
11	1.5	0.1	20	0.23	0.06	0.21	0.08
12	1.5	0.5	20	0.95	0.26	0.88	0.35
13	3.5	0.1	20	0.12	0.03	0.11	0.04
14	3.5	0.5	20	0.45	0.13	0.42	0.16
15	3.8	0.1	20	0.12	0.03	0.11	0.03
16	3.8	0.5	20	0.42	0.12	0.39	0.15



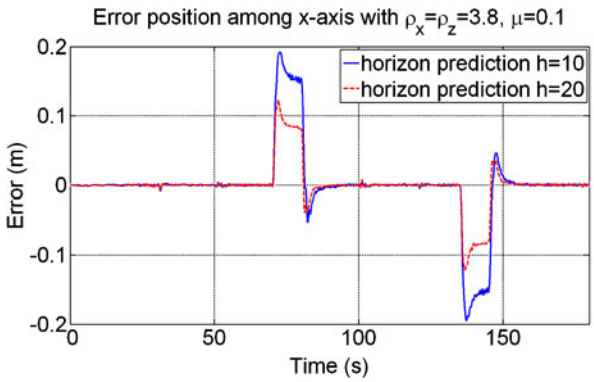
(a) *Position Error between leader and follower among x axis.* (b) *Position Error between leader and follower among z axis.*

Fig. 9 Simulation results with two different choices of horizon prediction. *Continue lines* are related to horizon prediction equal to 10 (simulation #3), *dashed lines* are related to prediction horizon equal to 20 (simulation #11)

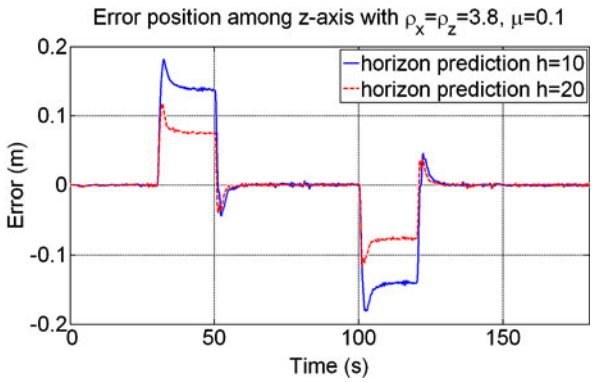


(a) *Position Error between leader and follower among x axis.* (b) *Position Error between leader and follower among z axis.*

Fig. 10 Simulation results with two different choices of horizon prediction. *Continue lines* are related to horizon prediction equal to 10 (simulation #5), *dashed lines* are related to prediction horizon equal to 20 (simulation #13)



(a) *Position Error between leader and follower among x axis.*



(b) *Position Error between leader and follower among z axis.*

Fig. 11 Simulation results with two different choices of horizon prediction. *Continue lines* are related to horizon prediction equal to 10 (simulation #7), *dashed lines* are related to prediction horizon equal to 20 (simulation #15)

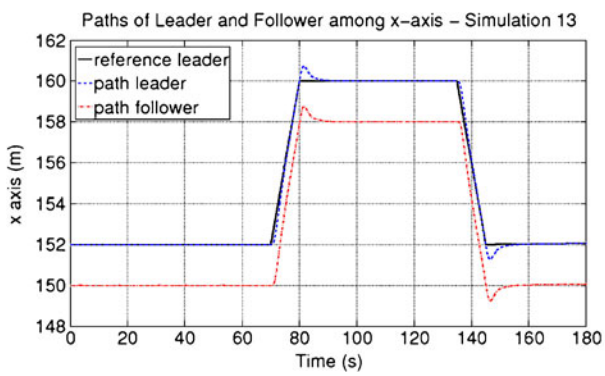
The shift of the leader aircraft from a waypoint to the next is carried out following a ramp reference. Clearly, the slope of the ramp determines the speed of the aircraft. After each shift, the leader lasts in the new waypoint for a time T_i, $i = 2, 4, 6, 8$. As mentioned before, the follower quadrotor, during the entire simulation, is forced to maintain a relative distance of 2 m from the leader among each of the two axes. The graphs shown above indicate a relative position error just about zero during intervals in which quadrotors remain in a waypoint and a non-zero relative position error (depending of the choice of ρ_x, ρ_z, μ and h parameters) during the routing from a waypoint to the next one. As an example, in Fig. 12a and b, the trajectories covered by the quadrotors during simulation 13 are reported. The position error at the generic t distant must be calculated as follows:

$$error = posLeader_i(t) - posFollower_i(t) - d \quad (24)$$

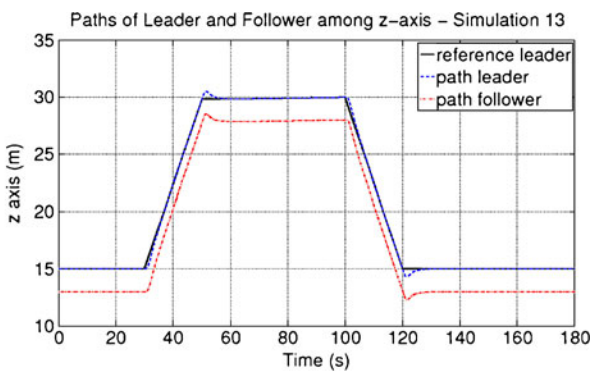
where d is the relative desired displacement between aircraft (in this case $d = 2$ m).

In accordance with the Eq. 24, relatively to the x-axis, at $t = 38.012$ s the error is:

$$\begin{aligned} error &= posLeader_x(38.012) \\ &\quad - posFollower_x(38.012) - d \\ &= 151.9991 - 149.9978 - 2 \backsimeq 0 \text{ m} \end{aligned} \quad (25)$$



(a) *Paths of Leader and Follower among x-axis.*



(b) *Paths of Leader and Follower among z-axis.*

Fig. 12 Simulation results of test #13. The *continuous lines* are the reference paths, the *dashed lines* are the leader paths and the *dotted-dashed* ones are the follower paths

while, at $t = 76.65$ s the error is:

$$\begin{aligned} error &= posLeader_x(76.65) \\ &\quad - posFollower_x(76.65) - d \\ &= 156.3859 - 154.2934 - 2 \backsimeq 0.09 \text{ m} \end{aligned} \tag{26}$$

Such values are in keeping with the simulation results.

The obtained result show that the ND-MPC gives good results also for formation control in the case of flying robots. The simulations confirm that increasing the prediction horizon and keeping fixed the weight coefficients, a substantial improvement in the relative position between leader and follower can be obtained. Clearly, the growth of the prediction horizon inevitably increases the value of the control effort, and with it the value of the functional.

6 Conclusion and Future Works

In this paper a modular framework for fast prototyping of cooperative autonomous was presented. A non linear MPC strategy to evaluate the performances of the framework has been considered. The modularity of this framework allows to quickly develop new modules without major changes of the already developed code. In addition it allows gray-box development of control systems and the ability to perform highly realistic simulations based on the most important physics engines currently available (Newton, PhysX, and so on). The virtual environment can be enhanced simply by integrating the three-dimensional models of objects. Development of control systems directly in Matlab/Simulink will provide the ability to generate code that can be downloaded directly on the hardware with tools like Real-Time Workshop. Thanks to the socket connection it is also possible to distribute the computational load across multiple computers on a network by providing an architecture formed by a computer in which the simulator runs and n computers with each one of them specialized for controlling of a modelled dynamic system.

Future work will be steered to improve the quality of the simulation by providing the ability to model sensors that allow a higher degree of realism, in particular Ultra Wide Band sensors for communication and localization among agents, in collaboration with Thales Italia. Moreover, new control algorithms based on a probabilistic representation of information will be analyzed.

Acknowledgement This work is developed in the context of ARTEMIS-JU EU Project R3-COP.

References

1. Player Stage Gazebo. http://playerstage.sourceforge.net/
2. Flightgear. http://www.flightgear.org/
3. Microsoft Robotics Developer Studio. http://www.microsoft.com/robotics/
4. Simplysim Simplycube. http://www.simplysim.net/
5. Physx. http://www.nvidia.com/object/physx-9.10.0513-driver.html
6. Newton. http://newtondynamics.com/forum/newton.php
7. X-plane. http://www.x-plane.com
8. Mancini, A., Cesetti, A., Iualé, A., Frontoni, E., Zingaretti, P., Longhi, S.: A framework for simulation and testing of uavs in cooperative scenarios. J. Intell. Robot. Syst. **54**, 307–329 (2009). doi:10.1007/s10846-008-9268-8
9. Frontoni, E., Mancini, A., Caponetti, F., Zingaretti, P.: A framework for simulations and tests of mobile robotics tasks. In: 14th Mediterranean Conference on Control and Automation, pp. 1–6 (2006)
10. Pnet. http://www.mathworks.com/matlabcentral/fileexchange/345
11. Sujit, P.B., George, J.M., Beard, R.W.: Multiple UAV coalition formation. In: American Control Conference 2008, pp. 2010–2015 (2008)
12. Yang, Z.J., Qi, X.H., Shan, G.L.: Simulation of flight control laws design using model predictive controllers. In: International Conference on Mechatronics and Automation, 2009. ICMA 2009, pp. 4213–4218 (2009)
13. Vaccarini, M., Longhi, S., Katebi, R.: State space stability analysis of unconstrained decentralized model predictive control systems. In: American Control Conference, 2006, p. 6 (2006)
14. Lawton, J.R.T., Beard, R.W., Young, B.J.: A decentralized approach to formation maneuvers. IEEE Trans. Robot. Autom. **19**(6), 933–941 (2003)
15. Xiaorui, Xi., Abed, E.H.: Formation control with virtual leaders and reduced communications. In: 44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05, pp. 1854–1860 (2005)
16. Ihle, I.-A.F., Jouffroy, J., Fossen, T.I.: Formation control of marine surface craft using lagrange multipliers. In: 44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference, pp. 752–758 (2005)

17. Dimarogonas, D.V., Kyriakopoulos, K.J.: Formation control and collision avoidance for multi-agent systems and a connection between formation infeasibility and flocking behavior. In: 44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05, pp. 84–89 (2005)
18. Dimarogonas, D.V., Kyriakopoulos, K.J.: On the state agreement problem for multiple unicycles. In: American Control Conference, 2006, p. 6 (2006)
19. Vaccarini, M., Longhi, S., Katebi, M.R.: Unconstrained networked decentralized model predictive control. J. Process Control **19**(2), 328–339 (2009)
20. Dunbar, W.B., Murray, R.M.: Distributed receding horizon control for multi-vehicle formation stabilization. Automatica **42**, 549–558 (2006)
21. Longhi, S., Monteriù, A., Vaccarini, M.: Cooperative Control of Underwater Glider Fleets by Fault Tolerant Decentralized MPC. In: 17th IFAC World Congress, pp. 16021–16026. Seoul, Korea (2008)
22. Vaccarini, M., Longhi, S.: Formation control of marine vehicles via real-time networked decentralized MPC. In: 17th Mediterranean Conference on Control and Automation, 2009. MED '09, pp. 428–433 (2009)