2020 International Conference on Unmanned Aircraft Systems (ICUAS)
Athens, Greece. September 1-4, 2020

Path-Following with a UGV-UAV Formation Considering that the UAV Lands on the UGV

Vinícius Pacheco Bacheti†, Alexandre Santos Brandão‡ and Mário Sarcinelli-Filho†

***Abstract*— This paper addresses the control of a formation composed of an unmanned ground vehicle (UGV) and an unmanned aerial vehicle (UAV). The formation navigates in a coordinated way, with the UAV following the UGV while keeping a relative pose with respect to the UGV as long as the UGV follows a specified path, so that the UAV can land on the UGV at any desired instant. This application is conceived in a context in which the UAV has finished a package delivery task and comes back to land over a platform used as a base for the package delivery operation. Experimental results are presented, which validate the approach adopted to accomplish such a task.**

I. Introduction

Unmanned aerial vehicles have been used as an autonomous robot to execute several tasks. Much of these tasks are related to the capability of collecting top view images, such as applications involving infrastructure inspection [1], [2], [3], agricultural inspection [4], [5], load transportation [6], [7], [8].

More recently applications involving multiple UAVs or even groups of UAVs and UGVs have been used in many applications, such as those in [9], [10], [11], [12], [13], [14], [15], just to mention a few, with gain in terms of cost effectiveness, time to task accomplishment, or even because of the task to be accomplished itself.

In such a context, this paper addresses the problem of the joint navigation of a formation composed of a UGV and a UAV (heterogeneous formation) to follow a predefined path, with the possibility that the UAV takes-off from the UGV or lands on it. Notice that this could be part of a task of package delivery by quadrotors, with the quadrotor departing from a mobile base to delivery the package, when close to the package destination, and/or landing back on it after delivering the package. The approach used is to consider the UAV and the UGV as a formation (a multi-robot system), and control the navigation of both vehicles in a coordinated way.

Notice that this work differs from those reported in [16] and [17], which deal with tasks of positioning and trajectory-tracking, respectively, whereas this work deals with path-following. In this sense it is worth mentioning that path-following implies that the vehicle (in this case the UGV-UAV formation) should follow a specified path, but the velocity can be freely set, whereas in positioning it is zero and in trajectory-tracking it is defined by the velocity of the trajectory. Therefore, considering the last step of a package delivery application, when the drone should land back in a terrestrial platform (a small truck, for instance), a path-following task fits better to the application than trajectory-tracking, as roads constraints, such as red lights and other cars, may affect the desired velocities in an unpredictable manner.

To discuss the topics involved, the paper is hereinafter split in five sections, starting with Section II, which describes the experimental setup adopted to run the experiments necessary to validate the proposal, and Section III, in which the formation UGV-UAV is characterized and described. In the sequel, Section IV describes the path-following controller proposed to guide the formation, and Section V shows results of some real experiments run using the *Pioneer 3-DX* and the *Bebop 2* vehicles. Finally, Section VI highlights the conclusions of the work.

II. The Experimental Setup

This section presents the experimental setup used to run the experiments discussed in Section V. The UGV adopted is the unicycle-like nonholonomic platform *Pioneer 3-DX*, from *Adept Mobile Robots*, and the UAV is the quadrotor *Bebop 2*[1], from *Parrot Drones SAS*, which are shown in Figure 1. The *Pioneer 3-DX* is a quite versatile small lightweight two-wheels differential drive robot used worldwide for education and research. In this work, in particular, it has a rectangular

Fig. 1: Pioneer 3-DX and Bebop 2 with the markers used for capturing their motion.

* This work was supported by CNPq-Brazil and FAPES-ES, Brazil

† V. P. Bacheti and M. Sarcinelli-Filho are with the Graduate Program on Electrical Engineering, Federal University of Espírito Santo, Vitória - ES, Brazil, where Mr. Bacheti is pursuing a Masters degree and Dr. Sarcinelli-Filho is a professor vinicius.bacheti@gmail.com, mario.sarcinelli@ufes.br

‡ A. S. Brandão is with the Department of Electrical Engineering, Federal University of Viçosa, Viçosa - MG, Brazil alexandre.brandao@ufv.br

[1] See https://www.parrot.com/us/drones/parrot-bebop-2-power-pack-fpv

platform over it to allow the landing of the UAV, as it is shown in Section V. In terms of its kinematics, the *Pioneer 3-DX* is a differential drive platform, whose control inputs are the linear velocity v and the angular velocity ω. Its kinematic model, the one used here, is well known from the literature, even when using a control point displaced of the virtual axle linking its driven wheels [18], [19].

As for the *Bebop 2* quadrotor, it is a rotorcraft aerial vehicle originally designed as a toy to be controlled by smartphones or tablets, mainly for entertainment purposes, for its quite good video generation capabilities. However, due to the implementation of a library available in ROS[2] (Robot Operating System), which connects an external computer with the onboard computer responsible for the low-level control of such a quadrotor, it is possible to use it as a platform to run experiments regarding navigation and guidance.

As for the model of the *Bebop 2* quadrotor adopted here, it is a rotorcraft aerial four-rotor vehicle that moves in response to changes in the speeds of each rotor. As a consequence of such changes, four forces arise, one associate to each propeller, and the inclination of the platform caused by such changes generates forces in the three directions of the Cartesian 3D space, and torques around the respective axes. The combination of such forces and torques makes the quadrotor fly in any desired direction, with the desired speed, provided that such velocities stay below the limits imposed by the manufacturer.

An important aspect of the *Bebop 2* quadrotor is that it is equipped with an autopilot, which is a low-level controller responsible for controlling the attitude of the vehicle. Such an autopilot allows the user to limit the pitch and roll angles (θ and ϕ), besides the linear velocities and rotational velocity around its Z axis. For applications in which no aggressive maneuver is adopted, such as in this work, the user can limit the pitch and roll angles to small values, thus allowing to approximate such values by zero values, meaning that the platform correspondent to the body of the quadrotor is always in the horizontal position, but is able to develop velocities in the directions X and Y of the 3D space [20], [21], enabling the user to send commands that are a four-entries vector composed by the linear speeds in the three axis and the angular speed around the Z-axis (yaw velocity).

This vector of commands is, then,

$$\mathbf{u} = \begin{bmatrix} u_{v_x} \\ u_{v_y} \\ u_{\dot{z}} \\ u_{\dot{\psi}} \end{bmatrix},$$

where u_{v_x} and u_{v_y} are velocity commands in the X- and Y-axis, $u_{\dot{z}}$ is the velocity command in the Z-axis and $u_{\dot{\psi}}$ is the command for the yaw velocity.

All these commands are sent to the vehicle as normalized values, ranging from 1, which corresponds to the maximum pitch and roll angles, the maximum vertical speed and the maximum rotational speed, when in the positive direction, to -1, which corresponds to the same variables when in the negative direction. As aforementioned, such parameters are configurable, through the internal autopilot of the *Bebop 2*. In this work the maximum value for the pitch and roll angles is 5°, the maximum vertical speed is $1m/s$ and the maximum rotation speed is $100^\circ/s$. As for their positive directions, they are, respectively, front, left, up and counter-clockwise.

It is important to establish these values, as the identified constants presented at the end of this section are directly dependent on them.

As a consequence of such simplifications, the simplified dynamic model proposed in [20] is adopted to represent the *Bebop 2* quadrotor, which is written as

$$\ddot{\mathbf{x}} = \mathbf{f}_1\mathbf{u} - \mathbf{f}_2\dot{\mathbf{x}}. \tag{1}$$

Such a model has been shown to be an acceptable approximation for low values of pitch and roll angles, as it is the case here. In such a model the vector $\ddot{\mathbf{x}}$ stands for the accelerations on the global frame, whereas $\dot{\mathbf{x}}$ is the vector of velocities in the frame of the vehicle, which are characterized as

$$\ddot{\mathbf{x}} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\psi} \end{bmatrix}, \text{ and } \dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \end{bmatrix}.$$

The matrices $\mathbf{f}_1$ and $\mathbf{f}_2$, by their turn, are given by

$$\mathbf{f}_1 = \begin{bmatrix} K_1\cos\psi & -K_3\sin\psi & 0 & 0 \\ K_1\sin\psi & K_3\cos\psi & 0 & 0 \\ 0 & 0 & K_5 & 0 \\ 0 & 0 & 0 & K_7 \end{bmatrix} \text{ and}$$

$$\mathbf{f}_2 = \begin{bmatrix} K_2\cos\psi & -K_4\sin\psi & 0 & 0 \\ K_2\sin\psi & K_4\cos\psi & 0 & 0 \\ 0 & 0 & K_6 & 0 \\ 0 & 0 & 0 & K_8 \end{bmatrix}.$$

As for the constants $K_1, \cdots, K_8$ that appear in $\mathbf{f}_1$ and $\mathbf{f}_2$, they were experimentally identified with the help of the high precision motion capture system *OptiTrack*[3], configured with eight cameras, to gather the required data. Once the data were gathered, the method presented in [22] was adopted to analyze it and obtain the parameter values. Figure 2 shows the robots in the middle of the working space (an indoor environment), as well as some of the cameras of the *OptiTrack* system around the room. Besides, markers are used to allow the *OptiTrack* system to identify the robots, which can be seen in Figures 1 and 2. The resultant parameter values are presented in Table I.

TABLE I: The values identified for the model parameters

K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8
0.8417	0.1823	0.8354	0.1710	3.966	4.001	9.8524	4.7295

[2]See https://www.ros.org/

[3]See https://optitrack.com/motion-capture-robotics/

Fig. 2: Pioneer 3-DX and Bebop 2 with their markers.

III. Formation Characterization

To characterize the UGV-UAV formation considered in this work the paradigm of virtual structure [23], [24], [25] is adopted. Such a structure is characterized as a straight line in the 3D Cartesian space, which is the line segment linking the two vehicles. According to this paradigm, a virtual robot is put on a reference point in the geometric element characterizing the formation. It can be the center of gravity of the geometric element characterizing the formation, such as in [26], [27], [18], which, in these works, would be the middle point of the line linking the UGV and the UAV. However, as the objective is to propose a control approach that allows the UAV to land on the UGV, the point of interest for control, in the present work, will be the extremity of the virtual line correspondent to the UGV, such as in [24]. Thus, the formation characterization is the one shown in Figure 3.

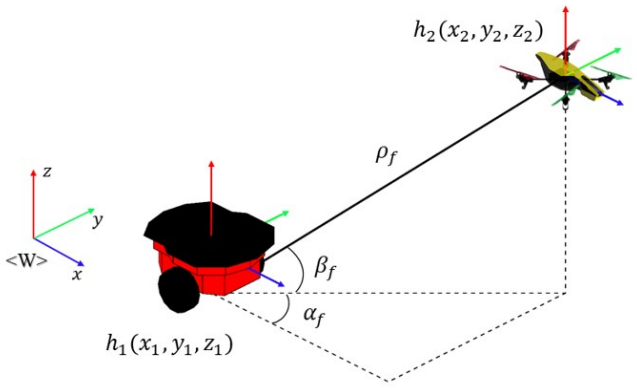

Fig. 3: Characterizing the UGV-UAV formation.

The variables describing the formation are also shown in Figure 3, where the index 1 stands for the UGV and the index 2 stands for the UAV. The variables correspondent to the positions of the UGV ($\mathbf{h}_1 = \begin{bmatrix} x_1 & y_1 & z_1 \end{bmatrix}^T$) and the UAV ($\mathbf{h}_2 = \begin{bmatrix} x_2 & y_2 & z_2 \end{bmatrix}^T$) are grouped in the vector $\mathbf{x} = \begin{bmatrix} \mathbf{h}_1^T & \mathbf{h}_2^T \end{bmatrix}^T = \begin{bmatrix} x_1 & y_1 & z_1 & x_2 & y_2 & z_2 \end{bmatrix}^T$, which describes the so called robot variables, whereas the variables grouped in the vector $\mathbf{q} = \begin{bmatrix} x_f & y_f & z_f & \rho_f & \alpha_f & \beta_f \end{bmatrix}^T$ are called formation variables. As previously mentioned, the virtual robot coincides with the UGV position, so $\begin{bmatrix} x_f & y_f & z_f \end{bmatrix}^T = \begin{bmatrix} x_1 & y_1 & z_1 \end{bmatrix}^T$. Thus, two different spaces are defined, namely the formation space (the cluster space in [26] and in [19]) and the robots space, which are connected by two transformations, a direct one, allowing getting $\mathbf{q}$ from $\mathbf{x}$, characterized as

$$\mathbf{q} = f(\mathbf{x}), \tag{2}$$

and a inverse one, allowing getting $\mathbf{x}$ from $\mathbf{q}$, characterized as

$$\mathbf{x} = f^{-1}(\mathbf{q}). \tag{3}$$

An important remark is that when considering the position of the UGV we are considering a value z_1, which is always zero in this paper. To consider the variable z_1, however, allows using this description for a formation of two UAVs, for instance, what caused the decision of keeping the position of the UGV as a 3D position.

IV. The Path-Following Controller for the Formation

The control structure adopted in this work is illustrated in Figure 4, which shows the different layers and how they interact to allow guiding the formation as a whole.

The goal of a path following task is to reach and follow a predetermined path with no time constraint. A path is defined by a parametric function $c(s) = (x(s), y(s), z(s)) \in \mathbb{R}^3$, where s is the curvilinear abscissa. As the formation in this paper is based on the UGV, $z(s)$ will always be constant and equal to zero. The path parametrization layer is responsible for this, in the proposed control structure. Moreover, once defined the path is static with respect to time, which allows the parametrization to be done prior to the navigation, thus out of the control loop. The control layer, however, still has to determine the point in the path that is the closest point to the robot and its distance, as this result will influence the behavior of the controller. Notice that this is because the position of the robot with respect to the path is time-dependent.

A. Control Layer

The nearest point between the path and the robot is defined as $c_n = \begin{bmatrix} c_{n_x} & c_{n_y} & c_{n_z} \end{bmatrix}^T$. The distance between c_n and $\begin{bmatrix} x_f & y_f & z_f \end{bmatrix}^T$ is defined as $\tilde{n}$, where $\mathbf{t}$ is the tangential unitary vector with respect to c_n and $\psi_{\mathbf{t}}$ is the orientation of $\mathbf{t}$ in the reference inertial system.

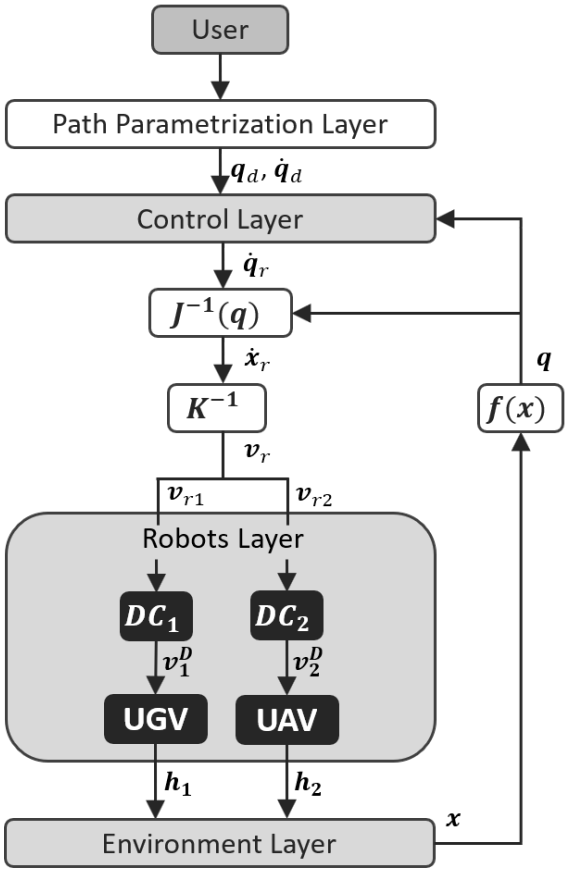

Fig. 4: Flowchart of the formation control system.

The control law for the formation, based on [24], is given by

$$\dot{\mathbf{q}}_r = \dot{\mathbf{q}}_d + \mathbf{L}_1 \tanh\left(\mathbf{L}_2 \tilde{\mathbf{q}}\right) \tag{4}$$

where $\mathbf{L}_1$ and $\mathbf{L}_2$ are diagonal positive definite saturation and gain matrices, respectively. As for the desired velocities $\dot{\mathbf{q}}_\mathbf{d}$ and the formation errors $\tilde{\mathbf{q}}$, they are

$$\dot{\mathbf{q}}_\mathbf{d} = \begin{bmatrix} \dot{x}_{f_d} \\ \dot{y}_{f_d} \\ \dot{z}_{f_d} \\ \dot{\rho}_{f_d} \\ \dot{\alpha}_{f_d} \\ \dot{\beta}_{f_d} \end{bmatrix} \text{ and } \tilde{\mathbf{q}} = \begin{bmatrix} \tilde{x}_f \\ \tilde{y}_f \\ \tilde{z}_f \\ \tilde{\rho}_f \\ \tilde{\alpha}_f \\ \tilde{\beta}_f \end{bmatrix}, \tag{5}$$

respectively. Also, the right hand side of (4) has two terms, the first one correspondent to the desired speed, essential in trajectory tracking, and the second one dealing with position errors.

In [24] this controller was used as the formation controller in a trajectory-tracking task, whereas in this paper it is proposed to use it in a path-following task. To do that, an approach based on [19] was implemented. As such, the desired velocities for the virtual mass-less robot correspondent to the point of control of the formation (in this case coincident with the point of control of the UGV) are not defined as the time derivative of the path. Instead, they are defined as a desired value tangent to the path, which corresponds to

$$\begin{bmatrix} \dot{x}_{f_d} \\ \dot{y}_{f_d} \end{bmatrix} = \begin{bmatrix} V_e \cos\psi_T \\ V_e \sin\psi_T \end{bmatrix}, \tag{6}$$

where z_{f_d} has been omitted, as the virtual robot navigates always on the XY-plane (where $z = 0$), and the formation variables $\begin{bmatrix} \rho_{f_d} & \alpha_{f_d} & \beta_{f_d} \end{bmatrix}^T$ have also been omitted as they bear no influence on the path-following strategy. As for V_e, it is the reference velocity input of the path-following controller, and is defined as

$$V_e = \begin{cases} 0 & \text{for } \tilde{n} > e_c, \\ V_d & \text{otherwise,} \end{cases} \tag{7}$$

where e_c is a threshold value selected for the position error, and V_d is the desired velocity of the virtual robot along the path.

In the first case, when the distance between the robot and the nearest point of the path is greater than the error e_c, V_e equals zero and, in turn, so do $\dot{x}_{f_d}$ and $\dot{y}_{f_d}$, which means that the path-following task of the virtual robot associated to the formation is changed to a positioning task, with the virtual robot going to or coming back to the path. On the other hand, as the virtual robot approaches c_n, the distance error $\tilde{n}$ will eventually be less than e_c, entering in the second case, where $V_e = V_d$. The virtual robot will then have a desired velocity tangential to the path, but the position error is still greater than zero, which means that there is still a residual value in the second term of the control law. This ensures that the virtual robot will converge to c_n in a smooth curve, almost tangential to the path. Ultimately, the position of the virtual robot will be equal to c_n, and it will follow the path with the desired speed V_d.

This work will focus on the last case, where the virtual robot is already over the planned path. Moreover, as the virtual robot is coincident with the UGV, all considerations made for it in this section are valid for the UGV. Finally, as V_d can be freely set regardless of the path configuration, some considerations can be made as to its feasibility. For instance, an acceptable navigation speed on a straight line or smooth curve might be too large for a sharp curve. Such considerations will be expanded further in the text, when describing the experiments run.

B. Interface Formation Space/Robots Space

In subsection IV-A the reference velocities for the formation ($\mathbf{q}_d$) were calculated. This, however, only describes how the formation should move, and not how each one of the robots in the formation should move. In order to determine the latter, the variables should be transformed from the formation space ($\dot{\mathbf{q}}_r$) to the robots space ($\dot{\mathbf{x}}_r$). This is done through the Jacobean of the inverse transformation in (3), and is defined as

$$\mathbf{J}^{-1}(\mathbf{q}) = \frac{(\partial \mathbf{x})}{(\partial \mathbf{q})}, \tag{8}$$

which, in turn, is given by

$$\mathbf{J}^{-1}(\mathbf{q}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & c_{\alpha_f} c_{\beta_f} & -\rho_f s_{\alpha_f} c_{\beta_f} & -\rho_f c_{\alpha_f} s_{\beta_f} \\ 0 & 1 & 0 & s_{\alpha_f} c_{\beta_f} & \rho_f c_{\alpha_f} c_{\beta_f} & -\rho_f s_{\alpha_f} s_{\beta_f} \\ 0 & 0 & 1 & s_{\beta_f} & 0 & \rho_f c_{\beta_f} \end{bmatrix} \tag{9}$$

where c and s represent the functions $\cos()$ and $\sin()$, respectively.

The transformation from the formation space to the robots space, when regarding the velocities the formation controller propagates to the robots, is defined as the time derivative of (3), which is given by

$$\dot{\mathbf{x}}_r = \mathbf{J}^{-1}(\mathbf{q})\dot{\mathbf{q}}_r, \tag{10}$$

where $\dot{\mathbf{x}}_r = \begin{bmatrix} \dot{x}_{1_r} & \dot{y}_{1_r} & \dot{z}_{1_r} & \dot{x}_{2_r} & \dot{y}_{2_r} & \dot{z}_{2_r} \end{bmatrix}^T$, is the formation controller reference in the robots space with respect to the reference inertial system. However, the commands sent to the robot must be in their own reference system, which is done taking

$$\mathbf{v}_r = \mathbf{K}^{-1}\dot{\mathbf{x}}_r, \tag{11}$$

where $\mathbf{K}^{-1}$ is a block diagonal matrix for which each block of the diagonal is the inverse Kinematics matrix of each one of the two robots in the formation. This way, $\mathbf{K}^{-1}$ is defined as

$$\mathbf{K}^{-1} = \begin{bmatrix} c_{\psi_1} & s_{\psi_1} & 0 & 0 & 0 & 0 \\ -s_{\psi_1} & c_{\psi_1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{\psi_2} & s_{\psi_2} & 0 \\ 0 & 0 & 0 & -s_{\psi_2} & c_{\psi_2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{12}$$

As for $\mathbf{v}_r$, it is the velocity reference generated by the formation controller in the robots own system, defined as

$$\mathbf{v}_r = \begin{bmatrix} \mathbf{v}_{r_1} \\ \mathbf{v}_{r_2} \end{bmatrix}, \tag{13}$$

where $\mathbf{v}_{r_1}$ is a vector representing the velocity reference for the first robot (UGV) along the three axis, and $\mathbf{v}_{r_2}$ represents the same for the second robot (UAV), in their own inertial reference systems.

C. UAV orientation controller

The mapping from the robots space to the formation space shown in (2) takes only the robots positions as arguments and, naturally, all the considerations made so far are restricted to them. Due to the nonholonomic restriction of the unicycle-like UGV adopted in this work, it will always face the direction of its movement, so its orientation is path-based. On the other hand, the UAV has full movement on the plane regardless of the direction it is facing, and could very well complete the whole formation task in any orientation whatsoever. Thus, its orientation task can be decoupled from its positioning task.

Depending on the task, the UAV orientation (ψ_2) may not be important, so no change has to be made. However, for any number of reasons that may not be so. If one wishes to make use of its vision capabilities it is interesting to be able to point it in a chosen direction. It can point in the same direction of the UGV (ψ_1) to detect obstacles or hazards, for instance.

For this reason, an orientation controller was introduced, defined as

$$\dot{\psi}_r = \dot{\psi}_{d_2} + L_3 \tanh\left(L_4 \tilde{\psi}\right) \tag{14}$$

where L_3 and L_4 are saturation and gain scalars, respectively, $\dot{\psi}_d$ is the desired rotational speed in the Z axis, and $\tilde{\psi}$ is the orientation error. This controller is similar to the one presented in (4), except that now it controls a single dimension, namely the orientation of the UAV.

This work focused on applications where the UAV points in the same direction as the UGV, thus the desired rotational speed and the orientation error were defined as

$$\dot{\psi}_{d_2} = \dot{\psi}_{d_1} \tag{15}$$

and

$$\tilde{\psi} = \psi_d - \psi_2, \tag{16}$$

where

$$\psi_d = \psi_1. \tag{17}$$

This controller ensures that the UAV will seek and maintain the same orientation of the UGV, the latter being dependent on the path. Therefore both vehicles will be oriented according to the tangent to the path.

In Subsection IV-B $\mathbf{v}_{r_2}$ was defined as a vector representing the Cartesian velocity reference for the second robot (UAV) along the three axis. Now, the rotational speed reference calculated in this subsection is appended to such a vector, and $\mathbf{v}_{r_2}$ is redefined as

$$\mathbf{v}_{r_2} = \begin{bmatrix} v_{x_{r_2}} \\ v_{y_{r_2}} \\ v_{z_{r_2}} \\ v_{\psi_{r_2}} \end{bmatrix}.$$

D. Dynamic Compensation

The controller so far considered has been based on a massless virtual robot and only kinematic effects have been taken into account. Even the orientation controller presented in Subsection IV-C implements a first order kinematic control. Initial experiments showed that the UGV presented good results with only the kinematic controller, whereas the UAV performed poorly. To improve the performance of the UAV when navigating in formation with the UGV, a dynamic compensation module, as proposed in [28], was added to consider the dynamic effects associated with the robot.

The compensator, based on the model presented in (1), is defined as

$$\mathbf{v}_2^D = \mathbf{K}_u^{-1}(\dot{\mathbf{v}}_{r_2} + \mathbf{L}_d(\mathbf{v}_{r_2} - \mathbf{v}_2) + \mathbf{K}_v\mathbf{v}_2), \qquad (18)$$

where $\mathbf{v}_2^D$ is the output of the dynamic compensator, $\mathbf{K}_u = \text{diag}(\begin{bmatrix} K_1 & K_3 & K_5 & K_7 \end{bmatrix})$ and $\mathbf{K}_v = \text{diag}(\begin{bmatrix} K_2 & K_4 & K_6 & K_8 \end{bmatrix})$ are diagonal matrices using the coefficients of the model presented in (1),$\mathbf{L}_d$ is a diagonal positive definite gain matrix, and $\dot{\mathbf{v}}_{r_2}$ is the time derivative of $\mathbf{v}_{r_2}$.

Such a compensator changes the dynamics of the velocity error to an asymptotically stable linear one (feedback linearization control technique [29]), guaranteeing that the error between the velocities $\mathbf{v}_{r_2}$ (the desired one) and $\mathbf{v}_2$ converges to zero asymptotically.

E. Environment Layer

After the compensation, the control signals are sent to the robots. They, in turn, execute the commands and their new positions are detected through the use of the *OptiTrack* motion capture system. Such a system is configured with eight cameras, some of which can be seen in Figure 2, in which one can perceive the cameras, the markers used to allow the cameras to detect the vehicles and the vehicles themselves. As Figure 4 shows, these positions are then fed back to the formation controller through (2), which generates the updated formation variables correspondent to the vector $\mathbf{q}$. Besides the positions of the two vehicles, measured through the *OptiTrack* system, the velocities of the vehicles are also necessary, to feedback the dynamic compensation module adopted to the UAV. Such values are determined by numerically deriving the position data. Finally, the control cycle is repeated, at the sampling rate of 0.1 seconds, till the navigation is accomplished.

V. Experimental Results

An overview of the experimental setup adopted to run the experiments to validate the control system here proposed is given in Figure 5. There it is sketched the connection of the three computers used, which are a Windows desktop running the *OptiTrack* motion capture system, a Linux notebook running the ROS nodes, and a Windows notebook running the control system, written in Matlab®. Complementing this overview, one can check Figure 2, which illustrates the UGV and the UAV during a flight, showing the markers attached to them to allow the *OptiTrack* cameras to get a good estimate of their positions.

A. UAV landing on a static UGV

The first experiment shown was conducted with a desired speed $V_d = 0.45\ m/s$, for a path defined as

$$\begin{bmatrix} x(s) \\ y(s) \\ z(s) \end{bmatrix} = \begin{bmatrix} 0.8\sin(\frac{2\pi s}{100})\ m \\ 1.2\sin(\frac{\pi s}{100})\ m \\ 0\ m \end{bmatrix}.$$

The desired formation during navigation is characterized as

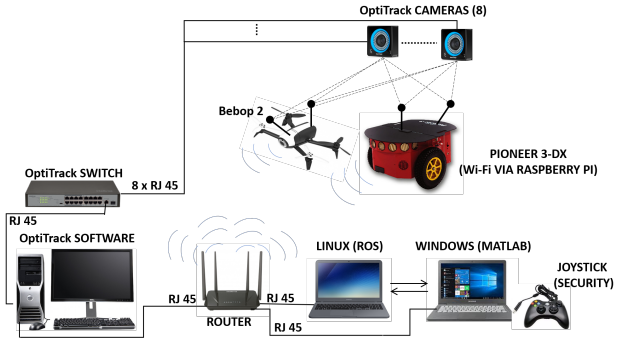

Fig. 5: An overview of the experimental setup used to run the validation experiments.

$$\mathbf{q}_{d_n} = \begin{bmatrix} c_{n_x} \\ c_{n_y} \\ c_{n_z} \\ 1.2\ m \\ 0\ rad \\ \frac{\pi}{2}\ rad \end{bmatrix},$$

and the desired formation during the landing procedure is characterized as

$$\mathbf{q}_{d_l} = \begin{bmatrix} c_{n_x} \\ c_{n_y} \\ c_{n_z} \\ 0.75\ m \\ (\psi_1 + \pi)\ rad \\ \frac{4\pi}{9}\ rad \end{bmatrix},$$

which centers the UAV on the platform on the top of the UGV. Also, at this stage $V_d = 0\ m/s$. Finally, the gains were tuned as

$$\begin{aligned} &\mathbf{L}_1 = \text{diag}\begin{bmatrix} 1.2 & 1.2 & 1 & 3 & 1.5 & 1.5 \end{bmatrix}, \\ &\mathbf{L}_2 = \text{diag}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}, \\ &L_3 = 1, \\ &L_4 = 1 \text{ and} \\ &\mathbf{L}_d = \text{diag}\begin{bmatrix} 2 & 2 & 1.8 & 5 \end{bmatrix}. \end{aligned} \qquad (19)$$

The task demanded that the formation should navigate for 30 seconds, and then the UAV should land on the static UGV.

The results are presented in Figures 6, 7 and 8. Besides, a video of the experiment can be found at the link `https://youtu.be/IPXWDFqnzSY`. As one can see from such figures, the formation managed to follow the proposed path. However, as previously commented, not all speeds are feasible to the robot in a given path. To make this comment clearer, another experiment was run, with the same gains and path, but with a slightly larger desired velocity $V_d = 0.5\ m/s$. The result, shown in Figure 9, is that the formation fails to follow the prescribed path, presenting quite big errors. The video of this experiment is available at the link `https://youtu.be/IPXWDFqnzSY?t=51`.

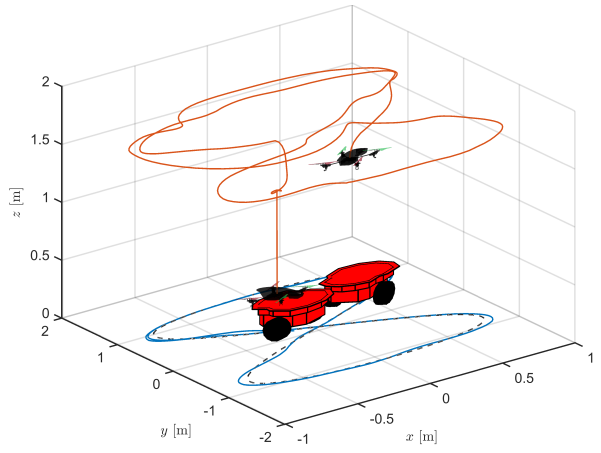

Fig. 6: Path traveled by the formation when $V_d = 0.45\ m/s$.

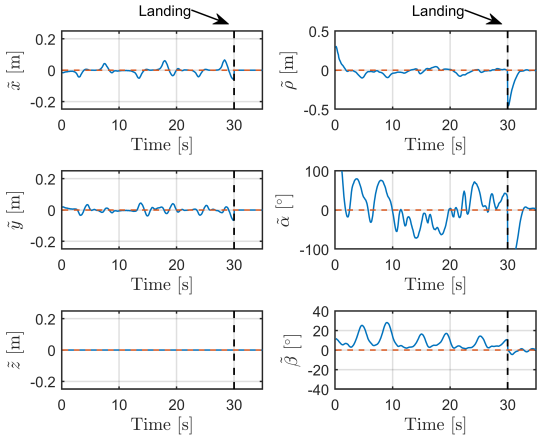

Fig. 7: Formation errors for the path-following task.

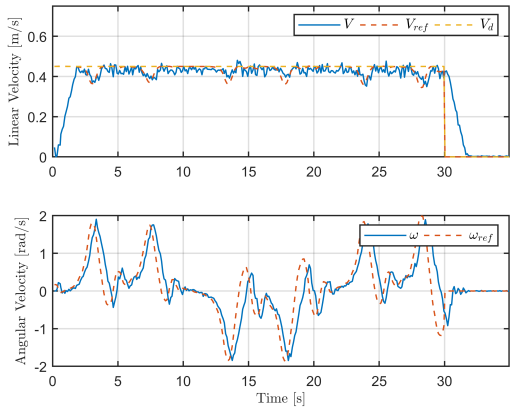

Fig. 8: UGV velocities for the path-following task.

To solve this problem, a strategy was adopted, which consists in reducing the velocity V_d when the angle of the tangent to the path is above a certain threshold (cautious navigation), which mimics the way a human driver behaves. To estimate such an angle the angular velocity of the UGV, $\dot{\psi}_1$, which coincides with that of the virtual robot correspondent to the

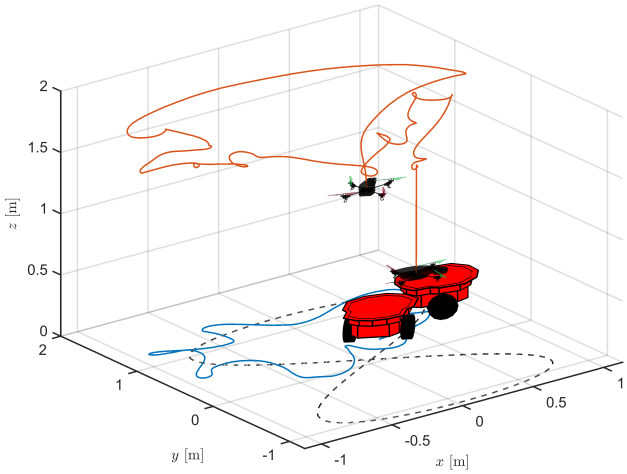

Fig. 9: Path traveled by the formation when $V_d = 0.5\ m/s$.

formation, is considered. Thus, the strategy is to consider

$$V_d = \begin{cases} \frac{V_{d_{max}}}{1+|\dot{\psi}_1|} & \text{for } |\dot{\psi}_1| > 0.5, \\ V_{d_{max}} & \text{otherwise,} \end{cases} \tag{20}$$

instead of a constant V_d, which means that the velocity along the path decreases to follow a sharp curve. After introducing such a strategy, the same path of the previous experiment was followed by the formation, keeping the gains, now using $V_{d_{max}} = 0.45\ m/s$, for the sake of comparison wih the first experiment. The result is summarized in Figures 10, 11 and 12, where one can see that the errors when following the proposed path are much smaller than when the correction of the desired velocity along the path is not included. Moreover, it is possible to observe that whenever the rotational speed of the UGV is high its desired linear speed drops accordingly. This experiment can also be understood watching the video available in `https://youtu.be/IPXWDFqnzSY?t=99`.

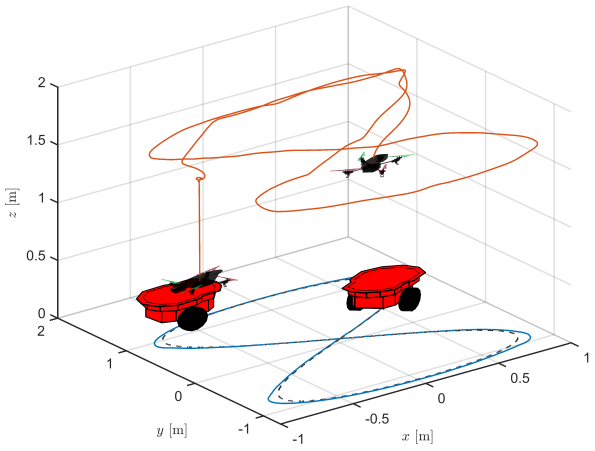

Fig. 10: Path traveled by the formation when $V_{d_{max}} = 0.45\ m/s$, with the strategy of changing V_d along the path.

To summarize, after defining the path to be followed and tuning the controller, the velocity $V_{d_{max}}$ should be defined, which is actually the maximum velocity along the path, not a

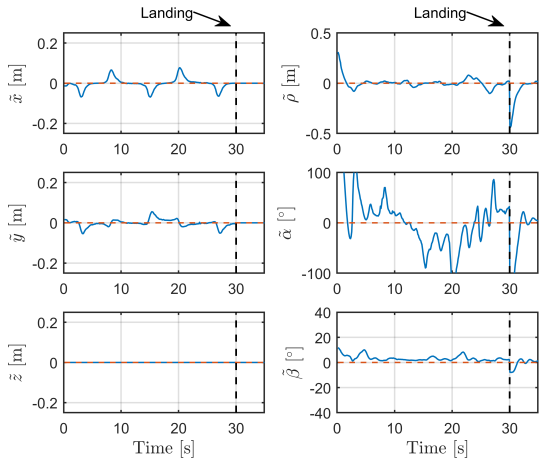

Fig. 11: Errors for the path-following task when correcting the velocity along the path.

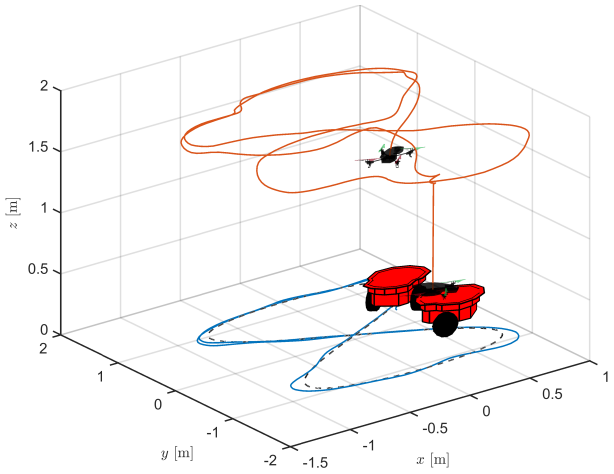

Fig. 13: Path traveled by the formation when $V_{d_{max}} = 0.65\ m/s$, with the strategy of changing V_d along the path.

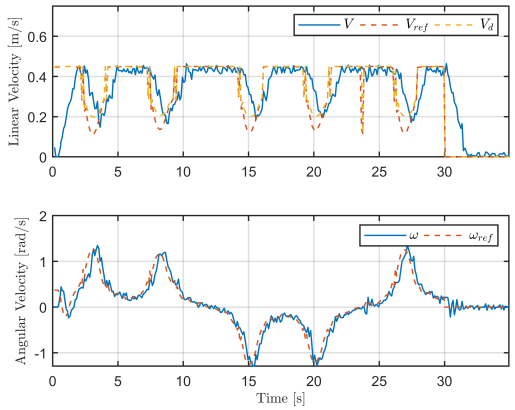

Fig. 12: Velocities of the UGV for the path-following task when correcting the velocity along the path.

constant velocity along the path. However, the velocity along the path can be constant and equal $V_{d_{max}}$, provided that the angular velocity of the UGV does not exceed the threshold established (the path is a smooth one).

Afterwards, the same experiment was run again, now considering a larger velocity of $V_{d_{max}} = 0.65\ m/s$, which is quite close to the upper limit for the velocity of the *Pioneer 3-DX* robot ($0.75\ m/s$). Figure 13 shows that the task was accomplished even considering this high velocity. The link `https://youtu.be/IPXWDFqnzSY?t=149` shows the video of this experiment.

B. UAV landing on and taking off from a moving UGV

The next experiment shows a navigation task involving the UAV taking off from a static UGV, navigating for some seconds and then landing on a moving UGV. After a while, the UAV takes off from the moving UGV and some seconds later the UGV stops moving and the UAV makes the final landing on it.

In order to do that, the desired formations $\mathbf{q}_{d_n}$ and $\mathbf{q}_{d_l}$ of the previous subsection were used, with $V_{d_{max}} = 0.50\ m/s$, or two third parts of the maximum UGV speed, for the navigation part and $V_{d_{max}} = 0.0\ m/s$ for the static part of the mission.

The navigation correspondent to such an experiment is shown in Figure 14. The navigation was split in three parts to get a clearer presentation of the results. The first part shows the beginning of the navigation, up to the moment the UAV lands on the UGV in movement. The robots are shown at the beginning and the end of each part of the experiment. It is possible to perceive that initially the UAV described an arc wider than that of the UGV, and very soon they are vertically aligned, with the UAV right over the UGV, as demanded by the configuration of the desired formation variables.

The second part starts with the beginning of the landed navigation and finishes when the UAV takes off. As in the first part, and also the third part yet to come, the sketch of the robots represent the beginning and the end of each part of the experiment. Moreover, a black continuous line can be seen in such parts, which represents the last second of the previous part, whose objective is to better contextualize the navigation. This part shows that the UAV landed cleanly, and the UGV continued following the required path.

Finally, the third part shows the navigation from the takeoff of the UAV up to its final landing on a static UGV. Again, a black trail is shown, representing the final second of the previous part. In the first moments it is clear that the UAV describes a much wider arc. It happens because while the UAV is still in its takeoff procedure, the UGV does not stop following along the path at the required velocity, which in turn increases the shape errors of the formation. As the shape variables of the formation are described in spherical coordinates, their concurrent correction, after the UAV finishes its takeoff, describes such an arc. Afterwards, the robots are aligned and finally the UAV lands on the top of a static UGV. The video available at the link `https://youtu.be/IPXWDFqnzSY?t=196` shows the complete experiment.

Figures 15 and 16 show the formation errors and UGV ve-

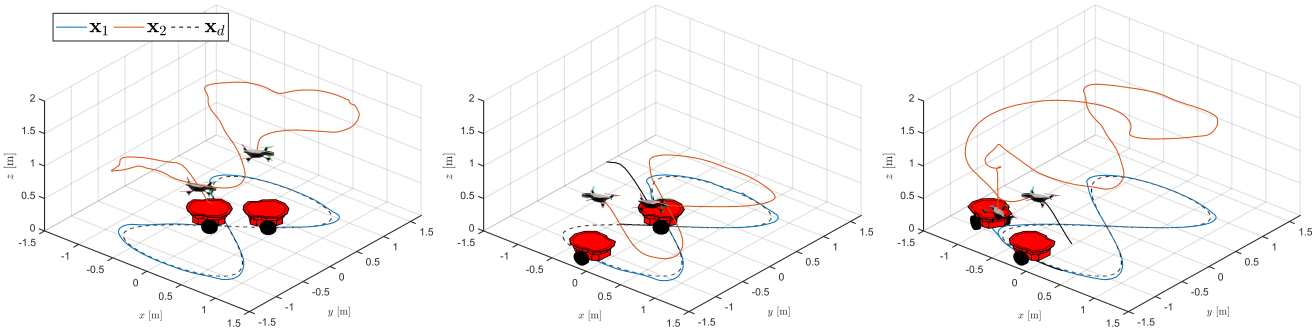

Fig. 14: Path traveled by the formation during the experiment, split in three parts.

locities in this experiment. The light blue boxes in Figure 15 represent the moments where the UAV is landed. In these moments, there are no errors associated with the shape of the formation, as shown on the right hand side of the figure.

Finally, one can have noticed that the errors associated with α_f are usually quite large. This happens because when $\beta_f = \frac{\pi}{2}$, as it is the case in $\mathbf{q}_{d_n}$, the UAV is in a singularity point, where small movements of the UAV cause a large variation of α_f, even though the UAV is approximately at the same position. This means that essentially, when at this singularity point, the values of α_f are meaningless to the formation, as any value will accomplish the task. On the other hand, if one considers the moments in which the UAV is preparing to land, where its desired formation is described by $\mathbf{q}_{d_l}$, the UAV is no longer at a singularity point and Figures 7, 11 and 15 show that, ahead of the line indicating landing, the controller brings the values of $\tilde{\alpha}_f$ back to values close to zero. Solutions for such a singularity condition will be explored further in future works.

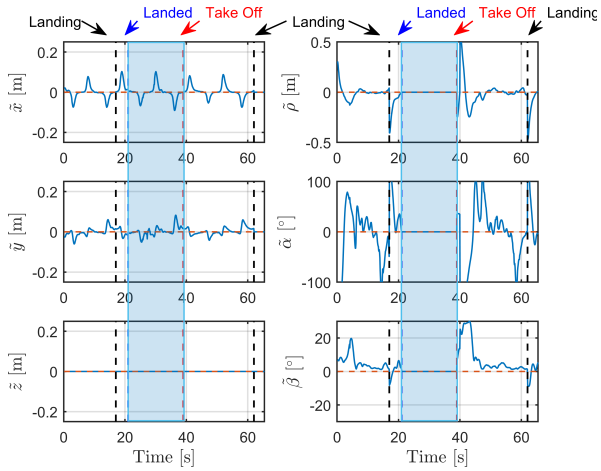

Fig. 15: Errors for the path-following task when the UAV lands and takes off from a moving UGV.

As a final remark, it is worthy mentioning that in all the experiments shown in this work the controller gains adopted are the same, which are those listed in (19).

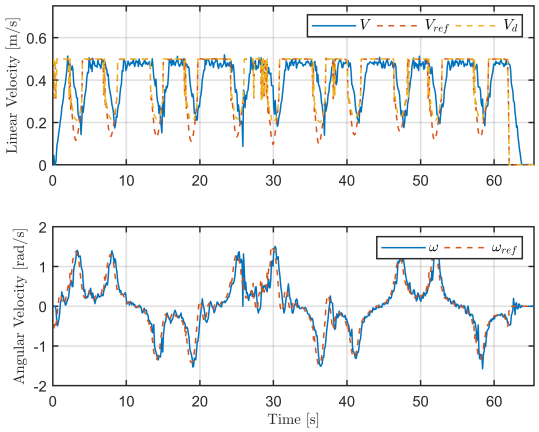

Fig. 16: Velocities of the UGV for the path-following task when the UAV lands and takes off from a moving UGV.

VI. Conclusion

In this paper a controller is proposed to guide a formation of a UGV and a UAV in path-following tasks. The objective is that the UAV can land on the UGV when it is stopped in a point along the path or is following such a path with a nonzero velocity. Such a controller involves also a correction of the UAV orientation, to guarantee that the UAV is always oriented as the UGV, if there is any demand in this sense. This is necessary because the UAV can be viewed as an omnidirectional vehicle, as it is a holonomic platform.

Besides, the proposed controller also allows adjusting the velocity along the path, based on an estimate of the curvature of the path, with is given by the magnitude of the angular velocity of the UGV, available from its odometry or by numerically differentiating its orientation, measured by an external system, such as the *OptiTrack* motion capture (used here). This way, the user defines the maximum speed along the path, not a constant one. In this sense, an important detail is that the proposed controller does not have any singularity in terms of the velocity along the path, which allows varying the velocity along the path in any way, including stopping the formation when necessary.

From the experimental results presented, it is possible to claim that the proposed controller is effective in guiding the

UGV-UAV formation to follow a given path, as expected from the theoretical analysis developed along the paper.

As future work, it is under analysis the possibility of using a null-space based version of this controller, to guarantee that the UAV will be in the desired position to land on the UGV, considering that the UGV changes its velocity along the path, in the presence of sharp curves or even to avoid an obstacle.

ACKNOWLEDGMENTS

The authors thank CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico, a Brazilian agency that supports scientific and technological development – and FAPES - Fundação de Amparo à Pesquisa e Inovação do Espírito Santo, an agency of the State of Espírito Santo that supports scientific and technological development – for the financial support granted to this work and for the scholarship granted to the first author. They also thank the Universidade Federal do Espírito Santo and the Universidade Federal of Viçosa, for supporting the development of this research.

REFERENCES

[1] S. Jordan, J. Moore, S. Hovet, J. Box, J. Perry, K. Kirsche, D. Lewis, and Z. T. H. Tse, "State-of-the-art technologies for uav inspections," *IET Radar, Sonar Navigation*, vol. 12, no. 2, pp. 151–164, 2018.

[2] B. W. Jiang, C. H. Kuo, K. J. Peng, K. C. Peng, S. H. Hsiung, and C. M. Kuo, "Thrust vectoring control for infrastructure inspection multirotor vehicle," in *2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA)*, April 2019, pp. 209–213.

[3] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48 572–48 634, 2019.

[4] C. Zhang and J. M. Kovacs, "The application of small unmanned aerial systems for precision agriculture: a review," *Precision agriculture*, vol. 13, no. 6, pp. 693–712, 2012.

[5] D. Murugan, A. Garg, and D. Singh, "Development of an adaptive approach for precision agriculture monitoring with drone and satellite data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 12, pp. 5322–5328, December 2017.

[6] I. Palunko, P. Cruz, and R. Fierro, "Agile load transportation: Safe and efficient load manipulation with aerial robots," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 69–79, September 2012.

[7] S. Dai, T. Lee, and D. S. Bernstein, "Adaptive control of a quadrotor uav transporting a cable-suspended load with unknown mass," in *53rd IEEE Conference on Decision and Control*, December 2014, pp. 6149–6154.

[8] I. H. B. Pizetta, A. S. Brandão, and M. Sarcinelli-Filho, "Control and obstacle avoidance for an uav carrying a load in forestal environments," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, Dallas, TX, USA, June 2018, pp. 62–67.

[9] L. E. Parker, *Multiple Robot Systems*, 2nd ed., ser. Springer Handbooks. Springer International Publishing, 2016, pp. 1335–1384.

[10] X. Dong, Y. Hua, Y. Zhou, Z. Ren, and Y. Zhong, "Theory and experiment on formation-containment control of multiple multirotor unmanned aerial vehicle systems," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 1, pp. 229–240, January 2019.

[11] I. H. B. Pizetta, A. S. Brandão, and M. Sarcinelli-Filho, "Cooperative load transportation using three quadrotors," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, Atlanta, GA, USA, June 2019, pp. 644–650.

[12] C. Masone, H. H. Bülthoff, and P. Stegagno, "Cooperative transportation of a payload using quadrotors: A reconfigurable cable-driven parallel robot," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. Daejeon, South Korea: IEEE, October 2016, pp. 1623–1630.

[13] H. Bai and J. T. Wen, "Cooperative load transport: A formation-control perspective," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 742–750, August 2010.

[14] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic uav and ugv system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[15] D. K. D. Villa, A. S. Brandão, and M. Sarcinelli-Filho, "Rod-shaped payload transportation using multiple quadrotors," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, Atlanta, GA, USA, June 2019, pp. 1036–1040.

[16] M. S. M. Moreira, A. S. Brandão, and M. Sarcinelli-Filho, "Null space based formation control for a uav landing on a ugv," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, Atlanta, GA, USA, June 2019, pp. 1389–1397.

[17] V. Ernandes-Neto, M. Sarcinelli-Filho, and A. S. Brandão, "Trajectory-tracking of a heterogeneous formation using null space-based control," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, Atlanta, GA, USA, June 2019, pp. 187–195.

[18] C. Z. Resende, R. Carelli, and M. Sarcinelli-Filho, "Coordinated path-following for multi-robot systems using the cluster space framework approach," in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, July 2014, pp. 332–337.

[19] C. Z. Resende, R. Carelli, T. F. Bastos-Filho, and M. Sarcinelli-Filho, "A new positioning and path following controller for unicycle mobile robots," in *2013 16th International Conference on Advanced Robotics (ICAR)*. IEEE, 2013, pp. 1–6.

[20] L. V. Santana, A. S. Brandao, M. Sarcinelli-Filho, and R. Carelli, "A trajectory tracking and 3d positioning controller for the AR.Drone quadrotor," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2014, pp. 756–767.

[21] L. V. Santana, A. S. Brandão, and M. Sarcinelli-Filho, "Navigation and cooperative control using the ar.drone quadrotor," *Journal of Intelligent Robotic Systems*, vol. 84, no. 1, pp. 327–350, 2016.

[22] M. C. P. Santos, C. D. Rosales, M. Sarcinelli-Filho, and R. Carelli, "A novel null-space-based uav trajectory tracking controller with collision avoidance," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 6, pp. 2543–2553, Dec 2017.

[23] M. A. Lewis and K.-H. Tan, "High precision formation control of mobile robots using virtual structures," *Autonomous Robots*, vol. 4, no. 4, pp. 387–403, October 1997.

[24] M. F. S. Rabelo, A. S. Brandão, and M. Sarcinelli-Filho, "Centralized control for an heterogeneous line formation using virtual structure approach," in *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*. IEEE, 2018, pp. 135–140.

[25] C. B. Low, "A flexible virtual structure formation keeping control design for nonholonomic mobile robots with low-level control systems, with experiments," in *Intelligent Control (ISIC), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 1576–1582.

[26] C. A. Kitts and I. Mas, "Cluster space specification and control of mobile multirobot systems," *IEEE/ASME Transactions on Mechatronics*, vol. 14, no. 2, pp. 207–218, April 2009.

[27] A. S. Brandão, V. T. L. Rampinelli, F. N. Martins, M. Sarcinelli-Filho, and R. Carelli, "The multilayer control scheme: A strategy to guide n-robots formations with obstacle avoidance," *Journal of Control, Automation and Electrical Systems*, vol. 26, no. 3, pp. 201–214, Jun 2015.

[28] M. C. P. Santos, C. D. Rosales, J. A. Sarapura, M. Sarcinelli-Filho, and R. Carelli, "An adaptive dynamic controller for quadrotor to perform trajectory tracking tasks," *Journal of Intelligent & Robotic Systems*, vol. 93, no. 1-2, pp. 5–16, 2019.

[29] H. Khalil, *Nonlinear Systems*, ser. Pearson Education. Prentice Hall, 2002.