A Path-Following Controller for a UAV-UGV Formation Performing the Final Step of Last-Mile-Delivery

VINÍCIUS PACHECO BACHETI1, ALEXANDRE SANTOS BRANDÃO2, AND MÁRIO SARCINELLI-FILHO3

1Graduate Program on Electrical Engineering, Federal University of Espírito Santo, Vitoria 29075-910, Brazil
2Department of Electrical Engineering, Federal University of Viçosa, Viçosa 36570-900, Brazil
3Department of Electrical Engineering, Federal University of Espírito Santo, Vitoria 29075-910, Brazil

Corresponding author: Mário Sarcinelli-Filho (mario.sarcinelli@ufes.br)

This work was supported in part by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), in part by the Fundação de Amparo à Pesquisa e Inovação do Espírito Santo (FAPES), in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), and in part by the Fundação de Amparo à Pesquisa de Minas Gerais (FAPEMIG), all from Brazil. The participation of Vinícius Pacheco Bacheti was supported by a scholarship paid by FAPES.

ABSTRACT

This work discusses the accomplishment of a path-following task by a formation of an unmanned aerial vehicle (UAV) and an unmanned ground vehicle (UGV). The idea behind such an application is that the UAV is returning after delivering a package, and should land on the UGV, its reference base, which continues following its route towards another delivery. A high level controller based on the virtual structure paradigm is designed to guide the formation, which also includes an obstacle avoidance algorithm to handle the possibility that the UGV, the reference for the formation, faces an obstacle in its path, such as another vehicle, for instance. After describing in details the controller designed and the obstacle avoidance algorithm, simulated and experimental results are shown and discussed, which validate the proposed controller.

INDEX TERMS

Heterogeneous formation, mobile robotics, multi-robot systems, obstacle avoidance, path-following controller.

I. INTRODUCTION

The use of unmanned aerial vehicles (UAVs) in tasks such as load transportation or manipulation has become a reality [1]. A single UAV or a group of them working cooperatively are options to accomplish load transportation/manipulation tasks [2]–[7]. In such a context, one activity that has received great attention is the so-called last-mile delivery [8], [9], which involves UAVs/drones delivering packages [10], and taking-off/landing from/on trucks navigating at a street network [9], [11]. Besides last-mile delivery, UGV-UAV formations have also been a feasible solution in warehouse inventory [12] and mapping using multiple robots [13], [14] (this last case calls attention to the recent news about Mars exploration, with a drone taking-off from a rover and landing back on it after acquiring images from their surroundings1).

As many countries are now approving regulations that opens the possibility of using small drones for package delivery, the last-mile delivery problem using autonomous navigation of UGV/UAVs has become an important research topic [9], [11], [15]. However, to accomplish such kind of task a high level of independence is necessary. For instance, when navigating a UGV-UAV formation should be able to react to obstacles in its surroundings, in order to avoid colliding with them. To do that autonomously, a controller to guide the formation along navigation is required, which should also allow the formation to take a new and safe path to avoid any obstacle appearing in the original path. This is the case in this work, which emulates the last step of a package delivery application, when the UAV should land back in a terrestrial

The associate editor coordinating the review of this manuscript and approving it for publication was Heng Wang.

1See https://mars.nasa.gov/mars2020/
moving platform (a small truck, for instance), after delivering a package. To be more specific, the step right before the landing of the UAV is considered, assuming that only the UGV faces obstacles, such as stopped or slow cars, for instance, in that circumstance. In this sense, the contribution of the paper is to show the feasibility of using the paradigm of virtual structure [16] to guide the UAV to land on a static or moving platform, which corresponds to the last step of the last-mile-delivery problem, i.e., the landing of the UAV on the UGV.

Regarding the virtual structure, it has become consolidated in the literature as a control approach for multi-robot systems, with many results already reported, adopting from ground robots [17] to surface vessels [18]. In most cases, however, homogeneous formations are considered. Some results regarding UAV-UGV formations have already been reported [11], [19], [20], most times considering positioning and/or trajectory-tracking tasks. However, an important aspect, when regarding UAV-UGV teams in package delivery applications, is that the ground vehicle is affected by constraints in road/avenues/streets, such as velocity limits, red lights and the presence of other cars, thus demanding the capability of modulating its velocity, an thus the velocity of the UAV, accordingly. To guarantee this possibility, this work adopts path-following as the motion control strategy, because it allows the modulation of the velocity of the formation according to the environment, which is not possible when regarding trajectory-tracking, for instance.

Therefore, this paper addresses the navigation of a UAV-UGV formation (an heterogeneous one) when following a predefined path, considering that the UAV should land on the UGV, which can be stopped or moving. The objective is to control a virtual structure associated to the formation, thus coordinating the movement of the two robots to guarantee that the UAV gets close enough to the UGV to land on it while the ground vehicle keeps moving. In addition, an obstacle avoidance approach, based on reactive local path-planning, is embedded in the proposed controller, to allow the formation to avoid obstacles in the ground. The premise is that the UGV, which coincides with the virtual robot characterizing the formation, has no previous information about obstacles, and scans its surroundings for threats. Whenever an obstacle is found, a new safe path is established around the obstacle and the UGV temporarily leaves the planned path, going ahead along a new safe path until leaving the obstacle behind. Then, as the navigation along the original path becomes safe again, the UGV resumes its original path. To implement such an obstacle avoidance strategy, an algorithm to deal with a single obstacle is designed, which is then extended to handle multiple obstacles.

Many approaches can be adopted to allow obstacle avoidance in mobile robotics. The simplest of them is to build a map of the environment, mark the positions of the obstacles and use a path-planning graph algorithm, such as the traditional Dijkstra algorithm [21], A* algorithm [22], and Bellman-Ford algorithm [23], [24], or even a more recent one, such as the one proposed in [25], based on genetic algorithm, just to mention some, in order to establish a free-of-obstacles path to be followed. Unfortunately, graph algorithms require high computational effort, not available in small vehicles. Furthermore, in many cases the robots do not have full knowledge of the environment, or the environment can change after the map building. A solution often used is to combine global, usually slow and of low-fidelity, and local, fast and of high-fidelity, planning, such as in the framework proposed in [26].

Other common approaches to implement obstacle avoidance are artificial potential methods [27]–[29], with its elegant mathematical analysis, neuro-fuzzy methods [30], [31] and genetic algorithms [25], [32]. However, a potential drawback of artificial potential and genetic algorithm methods are their aptness to get the robot trapped in local minima when navigating in environments with close obstacles or narrow channels [33]. To overcome such limitations, this work has chosen to focus on a reactive path planning approach, considering that the UGV has prior knowledge about the presence of obstacles in its path. Notice that in this case, considering the application addressed, only obstacles in the ground are considered, since the UAV flies close to the UGV, thus being free of most aerial obstacles.

Regarding path-following controllers, several works have been published, some of them in the last five years. In [34] and [35], for instance, $H_{\infty}$ robust control technique is adopted to control a single ground vehicle and a vessel, respectively, when following a path. In the second case, in addition to the $H_{\infty}$ robust control technique, adaptive output feedback linearization is also used. By their turn, in [36] and [37], both considering autonomous underwater vehicles, the techniques of backstepping (associated to a parameter update law neural networks) and model predictive control are adopted. Regarding path following control for multi-robot systems, [38] proposes a controller based on inverse kinematics for a group of ground vehicles, whereas [39] proposes a robust moving path following controller, including a first order sliding mode term, applied to a group of marine vessels. Other less recent articles on path-following control are available in the literature, but to the extent of the authors knowledge this is the first time a path-following controller is designed to guide a UAV-UGV formation in the last-mile-delivery problem.

The algorithm here designed to allow the UGV, and thus the whole formation, to avoid obstacles in the ground has the advantage of using only relatively simple linear algebra and demanding low computational effort, similarly to the proposal of [40]. There, a robot navigates in an environment with unknown obstacles and, as the obstacles are detected, it defines new safe paths around them. However, in spite of being quite robust, such a strategy involves both offline and online calculations, and a somewhat complex implementation, with final results quite similar to those obtained here. Complexity in terms of implementation is also an issue for the method proposed in [26], although allowing very fast navigation, theoretically speaking. Other similar work is the one reported in [41], which deals with path-following by a
formation. However, it adopts a linear controller, which is too dependent on the operation point around which linearization was performed. Besides this, its collision avoidance is focused on intra-formation collision, which is not an issue here, due to the nature of the formation robots (the UAV does not fly so low as to collide with the ground vehicle).

To discuss the topics involved in our proposal, the paper is hereinafter split in four sections, starting with Section II, which characterizes the formation used in this work as well as discusses the path following controller designed to guide the formation while following the planned path, and Section III, which describes in detail the obstacle avoidance algorithm proposed in connection with the path-following controller. In the sequel, Section IV shows the results of some simulations and a real experiment run using the unicycle mobile platform Pioneer 3-DX as the UGV of the formation and the quadrotor Bebop 2 as the UAV composing the formation. Finally, Section V highlights the main conclusions of the work.

II. THE PROPOSED PATH-FOLLOWING CONTROLLER

In this section, the virtual structure correspondent to the formation is characterized and described. The design of a path-following controller to ensure the accomplishment of the navigation is also discussed, as well as its main components and some auxiliary modules, such as a dynamic compensator adopted to reduce the velocity-tracking error for the UAV.

A. FORMATION CHARACTERIZATION

The characterization of the UGV-UAV formation considered in this work is based on the paradigm of virtual structure [11], [16], [42], where the virtual structure is the 3D straight line linking the two vehicles. According to such a paradigm, a virtual robot is put on a reference point in the geometric element characterizing the formation, which can be its center of gravity, such as in [17], [43], and [38]. For the UGV-UAV formation this would correspond to the point in the middle of the line linking the two vehicles. However, taking into account that the objective here is to propose a control system that allows the UAV to land on the UGV, the point chosen to place the virtual robot was the extremity of the virtual line defining the virtual structure, which coincides with the UGV, such as in [11]. Therefore, the formation is characterized as shown in Fig. 1, with the virtual robot associated to the formation being the Pioneer 3-DX robot itself.

The variables describing the formation are also shown in Fig. 1, with the indexes 1 and 2 standing for the UGV and the UAV, respectively. The variables correspond to the positions of the UGV \( \mathbf{h}_1 = [x_1, y_1, z_1]^T \) and the UAV \( \mathbf{h}_2 = [x_2, y_2, z_2]^T \) are grouped in the vector \( \mathbf{x} = [\mathbf{h}_1^T, \mathbf{h}_2^T]^T = [x_1, y_1, z_1, x_2, y_2, z_2]^T \), which describes the so-called robot variables, whereas the variables grouped in the vector \( \mathbf{q} = [x_f, y_f, z_f, \rho_f, \alpha_f, \beta_f]^T \) are called formation variables. As previously mentioned, the virtual robot coincides with the UGV position, so \( [x_f, y_f, z_f]^T = [x_1, y_1, z_1]^T \). Thus, two different spaces are defined, the formation space (the cluster space in [17] and in [38]) and the robots space, which are connected by two transformations: a direct one, allowing getting \( \mathbf{q} \) from \( \mathbf{x} \), characterized as

\[
\mathbf{q} = f(\mathbf{x}),
\]

and an inverse one, allowing getting \( \mathbf{x} \) from \( \mathbf{q} \), characterized as

\[
\mathbf{x} = f^{-1}(\mathbf{q}).
\]

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 with no modifications, for instance, which led to the decision to keep the position of the UGV as a 3D position.

B. THE PATH-FOLLOWING CONTROLLER

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. Notice that as the formation in this paper is based on the UGV, \( z(s) \) will always be constant and equal to zero. In addition, once defined the path is static with respect to time. However, the control layer still has to determine the point of the path that is closest to the robot and its distance, as this result will influence the behavior of the controller, since the position of the robot with respect to the path is time-dependent, due to the movement of the robot.

As mentioned, the path is not time dependent, so its definition can be done outside of the control loop, which corresponds to the Path Parametrization Layer of Fig. 2, where a flowchart of the whole control system adopted is shown.

The point of the path that is closest to the robot is defined as \( c_n = [c_{n_x}, c_{n_y}, c_{n_z}]^T \), which is calculated in each iteration of the control loop. The distance between \( c_n \) and \( [x_f, y_f, z_f]^T \) is

\[
\tilde{n} = \sqrt{(x_f - c_{n_x})^2 + (y_f - c_{n_y})^2 + (z_f - c_{n_z})^2}.
\]
where \( t \) is a unit vector tangent to the path at \( c_n \), whose orientation, in the inertial reference system, is

\[
\psi_t = \arctan \left( \frac{\frac{dy(s)}{ds}}{\frac{dx(s)}{ds}} \right),
\]

for \( x(s) = c_n \), and \( y(s) = c_n \).

The control law for the formation, similar to the one adopted in [11], is given by

\[
\dot{q}_d = q_d + L_1 \tanh(L_2 \dot{q})
\]

where \( L_1 \) and \( L_2 \) are diagonal positive definite saturation and gain matrices, respectively, \( \dot{q}_d \) represents the desired velocities, and \( \dot{q} = q_d - q \) corresponds to the formation errors. However, in [11] this controller was used to guide the formation in a trajectory-tracking task, whereas its usage in this work is to guide the formation in a path-following task.

In order to do that, an approach based on [38] was implemented. Therefore, the desired velocity for the virtual mass-less robot associated to the point of control of the formation (which coincides with the point of control of the UGV) is not defined as the time derivative of the path, which is null once the path has no time dependency. Instead, it is defined as a desired value tangent to the path, or

\[
\begin{bmatrix}
\dot{x}_{fd} \\
\dot{y}_{fd}
\end{bmatrix} =
\begin{bmatrix}
V_e \cos \psi_t \\
V_e \sin \psi_t
\end{bmatrix}.
\]

Note that \( z_{fd} \) has been omitted, as the virtual robot always navigates on the \( XY \)-plane (where \( z = 0 \)). The formation variables \( [\rho_{fd} \alpha_{fd} \beta_{fd}] \) have also been omitted, as they bear no influence on the path-following strategy. As for \( V_e \), it is the reference velocity input for the path-following controller, and is defined as

\[
V_e = \begin{cases} 
0 & \text{for } \dot{n} > e_c, \\
V_d & \text{otherwise,}
\end{cases}
\]

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 \( e_c \), \( V_e \) is equal to zero, which makes \( \dot{x}_{fd} \) and \( \dot{y}_{fd} \) also equal to zero, 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 \( \dot{n} \) 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 \( c_n \), meaning that it is following the path with the desired speed \( V_d \).

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. Thus, a slightly modified version of a curvature strategy presented in [15] was implemented, consisting in reducing the velocity \( V_d \) when the angle of the tangent to the path is above a certain threshold (cautious navigation). This intends to mimic the way a human driver behaves, and allows for a higher velocity to be set for the task than it would be possible if the velocity had to remain constant along the whole path.

In order to estimate such an angle, the angular velocity of the UGV, \( \psi_1 \), is considered. Thus, the strategy consists in considering

\[
V_d = \begin{cases} 
\frac{V_{d_{\text{max}}}}{1 + L_c |\psi_1|} & \text{for } |\psi_1| > 0.5, \\
V_{d_{\text{max}}} & \text{otherwise},
\end{cases}
\]

instead of a constant \( V_d \), which means that the velocity along the path decreases to follow a sharp curve. In such equation \( L_c \) is a gain constant to adjust the sensibility of the described strategy.

To summarize, after defining the path to be followed and tuning the controller, the velocity \( V_{d_{\text{max}}} \) should be defined, which is actually the maximum velocity along the path, not a constant velocity along the path. However, the velocity along the path can be constant and equal \( V_{d_{\text{max}}} \), provided that the angular velocity of the UGV does not exceed the threshold established (the path is a smooth one).
Going ahead in Fig. 2, even though the reference signals in the formation space have already been determined by (5), it is still necessary to calculate the control signals in the robots space. This is accomplished through the Jacobean of the inverse transformation in (2), which is defined as

\[
J^{-1}(q) = \frac{\partial \mathbf{x}}{\partial \mathbf{q}},
\]

which, in turn, results in

\[
J^{-1}(q) = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & c_{\beta_f} & -\rho_f s_{\beta_f} c_{\beta_f} & -\rho_f c_{\beta_f} s_{\beta_f} \\
0 & 1 & 0 & s_{\beta_f} & \rho_f c_{\beta_f} c_{\beta_f} & -\rho_f c_{\beta_f} s_{\beta_f} \\
0 & 0 & 1 & s_{\beta_f} & 0 & \rho_f c_{\beta_f} \beta_f \\
\end{bmatrix},
\]

where \(c_x\) and \(s_x\) stand for \(\cos(x)\) and \(\sin(x)\), 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 (2), which is given by

\[
\dot{x}_r = J^{-1}(q) \dot{q}_r,
\]

where \(\dot{x}_r = [\dot{x}_1, \dot{y}_1, \dot{z}_1, \dot{x}_2, \dot{y}_2, \dot{z}_2]^T\), 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 by taking

\[
v_r = K^{-1} \dot{x}_r,
\]

where \(K^{-1}\) 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. Thus, \(K^{-1}\) is defined as

\[
K^{-1} = \begin{bmatrix}
c_{\psi_1} & s_{\psi_1} & 0 & 0 & 0 & 0 \\
-s_{\psi_1} & c_{\psi_1} & 0 & 0 & 0 & 0 \\
a & a & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & c_{\psi_2} & s_{\psi_2} & 0 \\
0 & 0 & -s_{\psi_2} & c_{\psi_2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}.
\]

As for \(v_r\), it is the velocity reference generated by the formation controller in the robots own system, defined as

\[
v_r = [v_{r_1} v_{r_2}]^T,
\]

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

The controller so far considered has been based on a mass-less virtual robot and only kinematic effects have been taken into account. Initial experiments showed that the UGV presented good results navigating solely with 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 [44], was added to consider the dynamic effects associated with the robot.

The model used by the compensator, proposed in [45], is

\[
\dot{x} = f_1 u - f_2 \dot{x},
\]

which has been shown to be an acceptable approximation for low values of pitch and roll angles. In such a model the vector \(\dot{x}\) stands for the accelerations on the global frame, whereas \(\dot{x}\) is the vector of velocities in the frame of the vehicle. The compensator, based in [46], is then defined as

\[
v^0_r = K_u^{-1} (v_{r_2} + L_d (v_{r_2} - v_2) + K_1 v_2),
\]

where \(v^0_r\) is the output of the dynamic compensator, \(K_u = \text{diag}([K_1 K_2 K_3 K_4])\) and \(K_1 = \text{diag}([K_2 K_4 K_6 K_8])\) are diagonal matrices with the coefficients of the model, \(L_d\) is a diagonal positive definite gain matrix, and \(v_{r_2}\) is the time derivative of \(v_{r_2}\).

Such a compensator changes the dynamics of the velocity error to an asymptotically stable linear one (feedback linearization control technique [47]), guaranteeing that the error between the velocities \(v_{r_1}\) (the desired one) and \(v_2\) converges to zero asymptotically. A more detailed explanation of the model and how its parameters for the present vehicle were found can be seen in [46].

At last, an orientation controller for the UAV was added in order to make it face the same direction of the UGV. This has no immediate impact on the navigation, and is done for the sole reason of future works using the camera on the front of the flying vehicle. The complete details of the controller implemented are in [15], where further discussion about it is available.

### III. OBSTACLE AVOIDANCE ALGORITHM

This section discusses the algorithm developed in this work to allow the UAV-UGV formation to overcome obstacles found during the path-following task guided by the controller proposed. As a first step it will be discussed the avoidance procedure for a single obstacle. In the sequel the algorithm will be extended to handle multiple obstacles. In this last case, however, if the multiple obstacles are distributed in such a way that the formation can resume the original path being followed after overcoming an obstacle, then they can be dealt with as several single obstacles. Therefore, the algorithm for multiple obstacles is used just when the obstacles are close to each other. Furthermore, as in this work the obstacles are ground-based, the analysis will be focused on the virtual robot that characterizes the formation (remember that the formation control paradigm adopted is the one based on virtual structures [16], for which the entire formation is dealt with as a single virtual robot), which, for the formation considered, coincides with the UGV.

### A. AVOIDING A SINGLE OBSTACLE

When navigating along a path, a robot may face an obstacle blocking its way so that if the robot continues on its planned route a collision will occur. In such a situation, it is desirable
that the robot leave its original path and follow an alternative safe path, until it is safe to resume the original path.

The goal of the method presented here is to be a simple solution easily understood and implemented for this problem. As such, it is an algorithm that employs simple geometry in order to determine safe paths around the obstacle, and uses the same path-following controller discussed in Section II-B to navigate along this new path. Thus, the implemented strategy is an online path planning, not a controller-based one.

The proposed algorithm assumes that the position of each obstacle in the world is known as long as it has been detected by the robot. The zone of detection is a radius centered on the robot, whose position is given by \( x_f \), the position of the formation, and has the value of \( r_{det} \). Additionally, each obstacle will be modeled as a circumference with a radius large enough to encompass all of its physical boundaries and a safety zone, which will allow enough space for the body of the robot to navigate. This circumference is called the avoidance zone, and its radius is defined as \( r_{avd} \). It is assumed that \( r_{det} \) is larger than \( r_{avd} \) for all obstacles, as illustrated in Fig. 3.

![FIGURE 3. Obstacle avoidance situation with a single obstacle.](image)

Once the robot detects an obstacle, two special points in the original path \( c_{avd}(s) \) are defined. The first one is the first point in the path that intersects the circumference of radius \( r_{avd} \) centered on the center of the circumference that describes the obstacle, defined as \( x_{o_i} \) (for the \( i \)-th obstacle), whereas the second one is the second point of the original path which is over the same circumference. Notice in Fig. 3 that this means that the path being followed by the robot passes inside the safety zone of the obstacle, thus demanding a maneuver to avoid such obstacle. These two points are called \( c_{avd_{1_i}} \) and \( c_{avd_{2_i}} \), respectively. If none of such points exist, i.e., the obstacle is close enough to be detected by \( r_{det} \), but far enough such that \( r_{avd} \) does not touch the path, then the obstacle is not considered to be a threat to navigation and no action is taken. Furthermore, some approximations may be made in determining \( c_{avd_{1_i}} \) and \( c_{avd_{2_i}} \) depending on the resolution of the path, but common resolutions used are enough to ensure that such approximations have little impact.

As the distance between \( x_{o_i} \) and \( c_{avd_{1_i}} \) is \( r_{avd} \), and likewise for \( x_{o_i} \) and \( c_{avd_{2_i}} \), they fall exactly on the border of the circumference defined by the radius \( r_{avd} \) and centered in \( x_{o_i} \). The strategy adopted, then, consists in guiding the robot to temporarily abandon its original path \( c_{o}(s) \) and navigate tangentially to this circumference, thus preventing any collision with the obstacle. Moreover, as \( c_{avd_{1_i}} \) and \( c_{avd_{2_i}} \) both belong to \( c_{o}(s) \), the continuity between the paths is assured, i.e., the robot will meet its new path by following the original one, and once the avoidance maneuver is over, it will return to \( c_{o}(s) \) only after leaving the obstacle behind.

Of course, the robot might choose any direction, left or right, to navigate the new path. This choice could be random or follow any other criteria, such as the direction that the robot currently faces or the shorter path, for instance. For this work, the shorter path criterion was selected, and the new path \( c_{avd}(s) \), when regarding the example depicted in Fig. 3, is the one marked in the figure.

While at first it might seem a simple problem to determine \( c_{avd}(s) \) function as a circumference arc, this is somewhat complex because \( c_{avd_{1_i}} \), \( x_{o_i} \) and \( c_{avd_{2_i}} \) might be in any position and quadrant in the world reference frame \( W_{f} \). This leads to a complicated quadrant analysis of how to determine the curvilinear abscissa for the circumference arc. This is further compounded by the discontinuity present in the region of \([−\pi, \pi]\).

In order to tackle this problem, a new reference frame \( O_{f} \) is defined. The origin of this new frame is the center of the circumference describing the obstacle (\( x_{o_i} \)) and its \( x \)-axis points in the same direction of the line that connects \( c_{avd_{1_i}} \) and \( x_{o_i} \). This is accomplished through the homogeneous transformation matrix. Thus, the matrix \( H_{w}^{o} \) represents the position and orientation of the new frame \( O_{f} \) with respect to the world frame \( W_{f} \), and is written as

$$H_{w}^{o} = \begin{bmatrix} \cos \psi_{o} & -\sin \psi_{o} & 0 & x_{o_i} \\ \sin \psi_{o} & \cos \psi_{o} & 0 & y_{o_i} \\ 0 & 0 & 1 & z_{o_i} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$  (17)

where \( x_{o_i} = [x_{o_i} \ y_{o_i} \ z_{o_i}]^T \), \( c_{avd_{1_i}} = [x_{avd_{1_i}} \ y_{avd_{1_i}} \ z_{avd_{1_i}}]^T \), and \( \psi_{o} \) is the angle that \( x_{o_i} \) makes with the reference to \( c_{avd_{1_i}} \), defined as

$$\psi_{o} = \arctan\left(\frac{y_{o_i} - y_{avd_{1_i}}}{x_{o_i} - x_{avd_{1_i}}}\right).$$  (18)

However, to transform points in the world reference frame \( W_{f} \) to their corresponding points in the obstacle reference frame \( O_{f} \), the inverse transformation \( H_{w}^{o} \) is needed. Fortunately, due to the Homogeneous Transformation properties it is possible to state that

$$H_{w}^{o} = (H_{w}^{o})^{-1}.$$  (19)
The argument of the function should then start with $\psi$ of previous paragraph. Furthermore, reference frame, and thus

$$\begin{bmatrix} x_{o_i}^o \\ 1 \end{bmatrix} = H_w^o \begin{bmatrix} x_{o_i} \\ 1 \end{bmatrix},$$

(20)

with $x_{o_i}^o$ being the point correspondent to $x_{o_i}$ in the new reference frame,

$$\begin{bmatrix} c_{avd_j}^o \\ 1 \end{bmatrix} = H_w^o \begin{bmatrix} c_{avd_j} \\ 1 \end{bmatrix},$$

(21)

and

$$\begin{bmatrix} c_{avd_j}^o \\ 1 \end{bmatrix} = H_w^o \begin{bmatrix} c_{avd_j} \\ 1 \end{bmatrix},$$

(22)

where $c_{avd_j}^o$ and $c_{avd_j}^o$ are the points in the new frame that correspond to $c_{avd_j}$ and $c_{avd_j}$, respectively.

Due to this choice of frame, and the intrinsic characteristics of the points involved, in every situation it is possible to claim that

$$x_{o_i}^o = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \text{ and } c_{avd_j}^o = \begin{bmatrix} -r_{avd} \\ 0 \\ 0 \end{bmatrix}.$$  

(23)

Now the problem is greatly simplified as two of the points involved are constant, with only the last one, $c_{avd_j}^o$, presenting any variation. Moreover, the angle $c_{avd_j}^o$ makes with $x_{o_i}^o$, given by $\psi_{f}^o$, is right on top of the discontinuity, which means that the arc function will never pass through it, but rather always begin on it. Which value $\psi_{f}^o$ will assume, $\pi$ or $-\pi$, will depend on the direction desired for the arc, which will be explored shortly.

Fig. 4 shows the situation presented in Fig. 3 in relation to the reference frame $O_f$, and supports the claims made in the previous paragraph. Furthermore, $\psi_{f}^o$ is defined as the angle of $c_{avd_j}^o$ in relation to $x_{o_i}^o$.

The robot will always navigate a circumference arc beginning at $c_{avd_j}^o$ and ending at $c_{avd_j}^o$, when avoiding an obstacle. The argument of the function should then start with $\psi_{f}^o$ and end with $\psi_{f}^o$. Thus the curvilinear abscissa is defined as

$$s = \psi_{f}^o \frac{\text{inc}}{\psi_{f}^o},$$

(24)

where

$$\psi_{f}^o = \begin{cases} -\pi & \text{for } \psi_{f}^o < 0, \\ \pi & \text{otherwise}, \end{cases}$$

(25)

and inc is the increment of the function and is dependent on the resolution one wants for the path. This condition ensures that, if $\psi_{f}^o$ is negative, as in Fig. 4, the robot will follow the bottom arc, and if the angle is positive the robot will follow the top arc, thus always following the shorter path. Following this, the function that defines the shorter arc is

$$c_{avd_j}^o(s) = \begin{bmatrix} r_{avd} \cos s \\ r_{avd} \sin s \\ 0 \end{bmatrix},$$

(26)

where the $z$-coordinates are zero because the function is defined on the $xy$-plane.

The function defined, however, is with respect to the $O_f$ frame. In order to establish this function in the world coordinates, where our navigation is taking place, the inverse homogeneous transformation is applied, resulting in

$$\begin{bmatrix} c_{avd_j}^o(s) \\ 1 \end{bmatrix} = H_o^w \begin{bmatrix} c_{avd_j}^o(s) \\ 1 \end{bmatrix}. $$

(27)

Thus, as the robot encounters an obstacle when following its original path, the intersection of the path with the avoidance zone of the obstacle is determined. A new avoidance path will then be calculated once by the algorithm, beginning on the original path and following along the limits of the avoidance zone, on the shortest path available. When the avoidance path ends, it connects with the original path, and only when the robot reaches this point the obstacle is considered as left behind, and the robot resumes its navigation following the original path.

### B. AVOIDING multiple OBSTACLES

The algorithm described in the last section covers all the cases of single obstacles that the robot may find in its navigation along a path defined in the real world. However, what would happen if the robot faced a new obstacle during its navigation through the temporary path generated to avoid a previous obstacle? In the last section, one of the parameters used was $c_{avd_j}$, which provided continuity between the original path and the new one. However, now that the robot is not on the path, it makes no sense to use $c_{avd_j}$. Thus, a slightly modified version of the algorithm of the previous subsection is applied when the robot faces a new obstacle whilst avoiding a previous one. Notice that this is a real possibility when there are multiple obstacles in the navigation environment.

It will be assumed that the robot is facing a new obstacle $x_{o_{i+1}}$ when the distance between the robot, whose current position is $x_f$, and the center of the circumference characterizing the new obstacle, $x_{o_{i+1}}$, becomes smaller than the distance between $x_f$ and $x_{o_i}$, the center of the circumference characterizing the previous obstacle. Then, new parameters will be calculated and a procedure similar to the previous one will be run. Fig. 5 illustrates the situation and the relevant
parameters, considering two close obstacles. As previously mentioned, if the two obstacles were not close to one another, they would be dealt with as single obstacles.

The detection region encompassed by $r_{det}$ has been omitted for clarity. Another detail about the figure is that the robot has not yet detected the second obstacle when it starts avoiding the first one. Indeed, it will detect the second obstacle after entering the temporary path adopted to avoid the first obstacle, characterized as $c_{avd}(s)$. Moreover, the points $c_{avd_{i+1}}$ have been determined in a way similar to the one in the previous subsection, with the important difference that $c_{avd_{i+1}}$ is determined over the path the robot is currently following, i.e., $c_{avd}(s)$, and not over the original path $c_r(s)$. The goal is then to determine $c_{avd_{i+1}}(s)$, which will be the new safe path that the robot will follow, as $c_{avd}(s)$ has been compromised due to the new obstacle.

However, as one can see in Fig. 5, were the algorithm for multiple obstacles to use the same criterion previously adopted, the shortest path, the arc chosen would take the robot towards the previous obstacle. This means that it is not guaranteed that the algorithm should always choose the shortest path. It can take the largest one, as illustrated in the figure. As one can easily verify, if the second obstacle were moved upwards and placed horizontally above the first one, then the robot would really take the shortest path when avoiding the two obstacles. It is also easy to verify that one path (short or long) will always lead the robot to the previous obstacle, while the other will be safe for it.

In order to solve the problem of which arc should the robot take, a new criterion was adopted. After following the same procedure shown in the last subsection the algorithm calculates the shortest arc function, now identified as $c_{avd_{i+1}}(s)$, with the superscript $s$ on top of $avd$ to identify it as the short path.

To calculate the longer arc, the analysis remain the same, with the difference that now the curvilinear abscissa will be the complement of the previous one. This means that, for the long arc (24) remains the same, but (25) will be modified to

$$
\psi^L = \begin{cases} 
\pi & \text{for } \psi^L < 0, \\
-\pi & \text{otherwise}, 
\end{cases}
$$

which ensures that $c_{avd_{i+1}}(s)$ describes the longer arc, defined as

$$
c_{avd_{i+1}}(s) = \left[ \begin{array}{c}
 r_{avd} \cos s \\
r_{avd} \sin s \\
0
\end{array} \right],
$$

and

$$
\begin{bmatrix}
c_{avd_{i+1}}(s) \\
1
\end{bmatrix} = H_w^c \begin{bmatrix}
c_{avd_{i+1}}(s) \\
1
\end{bmatrix}.
$$

After defining both, the short and the long paths, the algorithm will calculate the distance of each of their points to the previous obstacle $x_{avd_i}$, and the minimum distance of each path will be identified as $d_{min_l}$ and $d_{min_s}$, for the short and long paths, respectively. Then, the new safe path will be defined as

$$
c_{avd_{i+1}}(s) = \begin{cases} 
c_{avd_{i+1}}(s) & \text{for } d_{min_l} > d_{min_s}, \\
c_{avd_{i+1}}(s) & \text{otherwise}. 
\end{cases}
$$

In other words, the path chosen will be the one farthest from the previous obstacle.

Furthermore, although this discussion focused on two obstacles, whenever the robot finds a new obstacle on its way, the robot will take the shorter path, as $c_{avd}(s)$ has been compromised due to the new obstacle.

IV. RESULTS

In order to test whether the proposed algorithm is a valid solution to the obstacle avoidance problem, simulations were run. After successful simulations, real world experiments were performed to further check the efficiency of the algorithm and its robustness. This section presents and discusses the results obtained in the simulations and the real experiments.

A. SIMULATIONS

The simulations goal is to validate the strategy proposed to accomplish obstacle avoidance. To achieve this, eight scenarios were devised to show the main challenges and variations of obstacles that the formation may face. The desired velocity of the formation in each case was $V_{d_{max}} = 0.4 \text{ m/s}$ with the gains tuned to

$$
L_1 = \text{diag} \left[ 1.2 \ 1.2 \ 1.3 \ 1.5 \ 1.5 \right],
$$

$$
L_2 = \text{diag} \left[ 1 \ 1 \ 1 \ 1 \ 1 \right],
$$

$$
L_3 = 1,
$$

$$
L_4 = 1,
$$

$$
L_d = \text{diag} \left[ 2 \ 2 \ 1.8 \ 5 \right] \text{ and }
$$

$$
L_c = 1.
$$

The desired formation in given by

$$
q_d = [c_{nx}, c_{ny}, c_{n\theta}, 1.5 \text{ m}, 0 \text{ rad}, \theta, \text{rad}]^T,
$$

FIGURE 5. Obstacle avoidance situation with multiple obstacles.
for whatever path $c(x)$ that is currently being followed in each scenario.

As for the obstacle avoidance algorithm parameters, the radii of detection and of the avoidance zone were defined as $r_{\text{det}} = 1.2 \text{ m}$ and $r_{\text{avd}} = 0.8 \text{ m}$. One should notice, however, that a zone of avoidance should be tied to each obstacle and that it might be different when regarding different obstacles. Although this is effectively true and easy enough to be handled by the algorithm as long as all $r_{\text{avd}}$ are smaller than $r_{\text{det}}$, for the sake of simplifying the discussion it was deemed better to use a single value for all obstacles.

The scenarios devised are shown in Fig. 6. The first column depicts the cases where the formation only encounters one obstacle at a time, i.e., even in the cases where there are multiple obstacles in the navigation, they are far enough apart that they may be treated separately. These cases will be discussed first, and then the second column, where the formation faces multiple obstacles at a time, will be discussed after.

The first figure shows the formation facing just a single obstacle, represented by the red circle, when following a straight path along the $x$ axis. The UGV starts from a point on top of the desired path whereas the UAV starts from a point at the UGVs right. They quickly converge to the desired formation, avoid the obstacle while maintaining the formation, and reach the end of the path.
It is readily apparent that the avoidance path is the shortest arc around the obstacle, as the center of the obstacle is below the path \((y_o < 0)\), and the formation navigates through positive values of \(y\). In addition, while it may seem that the value of \(r_{end}\) is unnecessarily large for the size of the obstacle, one has to keep in mind that only the control point of the UGV is being shown. In reality the robot has a non-zero size and it has to be taken into account, so the safe zone is made large enough to allow the whole body of the robot to navigate (this will become clearer in the experiments).

The second scenario is much like the first, except now there are two obstacles, one below and one above the path. Again, the algorithm is shown to yield the shortest arc of circumference around each obstacle for the formation to continue following the path. In other words, for the obstacle that is below the path the formation navigates above it, and for the obstacle that is above the path, the formation navigates below it.

The third scenario shows a more complex path where the algorithm has to determine which is the shortest path around the obstacles. The very first obstacle is an interesting case: at a first glance it would be better for the formation to steer to the left (in relation to where the formation is currently facing, which is downwards) as the obstacle is to the right of the path. However, after the obstacle the path makes a right turn, and, because of that, the shorter arc is actually to the right of the obstacle. This means that the algorithm is shown to be performing quite well in this somewhat complex and generic case. The other obstacles are similarly avoided.

So far, all the cases discussed have depicted the formation avoiding all the obstacles it faced by leaving the path, circumventing the obstacle following the shortest circumference arc and returning back to the original path beyond the danger and at a safe distance. This is in accordance to the expected behavior of the algorithm for single obstacles (or obstacles far enough in relation to the others).

Now, the cases presented in the second column, where multiple obstacles appear in the path followed by the formation close to one another, require the use of the extended algorithm. The first scenario, shown in the upper-right figure, depicts two obstacles in close proximity. Much like its single-obstacle counterpart, when facing the first obstacle, the formation avoids it using the shortest circumference arc available. However, in the middle of the avoidance maneuver, the robots face a second obstacle. Then, if the formation chose the shortest arc once more, it is easy to see that it would collide with the first obstacle. In such case, it is shown that the algorithm chooses the longest circumference arc, because it is the safest one. After that, the formation completes its navigation.

In the second scenario of multiple obstacles (middle figure at the right column), the formation has to deal with three close obstacles. Now, it is possible to observe that in each case, the robot followed the safest arc. The choices will depend on the relative positions of the obstacles, formation and path, as discussed, but it is clear that the algorithm yielded the right ones.

Finally, the last scenario shows two clumps of obstacles in close proximity. To avoid the first clump, the formation initially follows the shortest arc around the first obstacle and then, when faced with the next obstacle it chooses the longest arc to return to the path. Midway through its navigation, the formation is faced with the third obstacle of the clump, and it readily changes direction to avoid it. At the second clump, a similar process happens and the formation is able to navigate around it.

Thus, it has been shown that the algorithm will behave as expected from the theoretical analysis. It will avoid the first obstacle it sees following the shortest circumference arc and, upon facing a new obstacle, it will follow the safest circumference arc around the obstacle, as many times as necessary to return to the original path.

These simulation results show that the proposed strategy accomplishes the obstacle avoidance task in several different scenarios with unique challenges, indicating that it can deal with generic obstacle distributions in the workspace of the formation.

### B. REAL WORLD EXPERIMENT

This section presents and analyzes a real experiment in which the UAV-UGV formation should follow a path correspondent to a lemniscate of Bernoulli, along which it faces some obstacles. The experiment was run using the setup described on Fig. 7. As one can see in the figure, the UGV adopted was the nonholonomic unicycle Pioneer 3-DX mobile platform, and the UAV is the Bebop 2 quadrotor. To measure the vehicle positions of the vehicles along their navigation the motion capture system OptiTrack was used, configured with eight cameras spread around the room. In order to communicate with the robots the Robot Operating System - ROS was used. As for the high-level control code, it was run using Matlab. The communication between ROS and Matlab was done through a proprietary Matlab library, which simulates the necessary ROS nodes and topics. The obstacles were represented by physical objects, which also had their positions captured using the OptiTrack system.

**FIGURE 7.** An overview of the hardware setup used to run the experiment.
The desired formation velocity, the gains and the parameters of the obstacle avoidance algorithm were configured with the same values used in the simulations. The path to be followed was defined as

\[
\begin{bmatrix}
  x(s) \\
  y(s) \\
  z(s)
\end{bmatrix}
= \begin{bmatrix}
  -2.0 \sin\left(\frac{\pi s}{180}\right) \text{ m} \\
  1.2 \sin\left(\frac{2\pi s}{180}\right) \text{ m} \\
  0 \text{ m}
\end{bmatrix},
\]

(34)

As for the desired formation, it was established as being

\[
\mathbf{q}_{d_h} = \begin{bmatrix}
  c_{n_x} c_{n_y} c_{n_z} \\
  1.5 \text{ m} \\
  0 \text{ rad} \\
  \frac{\pi}{2} \text{ rad}
\end{bmatrix}^T,
\]

(35)

for the part of the experiment in which the two vehicles are navigating as a formation, and

\[
\mathbf{q}_{d_l} = \begin{bmatrix}
  c_{n_x} c_{n_y} c_{n_z} \\
  0.8 \text{ m} \\
  (\psi_1 + \pi) \text{ rad} \\
  4\pi \text{ rad}
\end{bmatrix}^T,
\]

(36)

for the part of the experiment in which the UAV is preparing to land on the UGV.

The task to be accomplished by the formation was to follow the path, reacting to any obstacle appearing in its way. During the first lap no obstacles were placed, in order to establish the baseline for normal navigation. In the second lap, two obstacles were placed in the way of the formation, far from each other. The objective was to show the algorithm working on avoiding single obstacles. Finally, in the third and last lap of the formation along the lemniscate of Bernoulli curve, the formation faced three obstacles placed near each other, in positions randomly selected along the navigation. The operation of the path-following controller and the obstacle avoidance algorithm can be watched in the video of the full experiment, which can be seen at the link https://youtu.be/KfihOr4xmE.

The results of the whole experiment are synthesized in Fig. 8, where, in order to make easier to understand the performance of the proposed system, the navigation was split into three parts, albeit being just a single experiment, each part consisting of a lap around the path.

In the first lap the two robots quickly converge to the positions correspondent to the establishment of the desired formation, and then they follow the path without problems, as depicted in the first subfigure. In the second lap, the formation faces an obstacle just about the second turn-right maneuver, as depicted in the second subfigure. As one can see in the figure, the formation avoids the obstacle, following a temporary safe path. Afterwards, just after turning left, a new obstacle forces the formation to follow a new safe path generated by the obstacle avoidance algorithm, which results in the avoidance of the obstacle.

In the last part of Fig. 8, the third lap was supposed to start at the same spot where the second lap finishes in the previous figure. However, as this was an obstacle free zone and the landing happened at this region, the beginning of the third lap was omitted, and a delayed starting point is shown in the graph. The formation starts following the lemniscate of Bernoulli in the third lap normally until facing three obstacles, just before turning left the first time. Then it uses the safe paths generated by the obstacle avoidance algorithm and navigates safely to a point in the path beyond the obstacles, from where it continues following the prescribed path.

Finally, at the end of the navigation, the UAV lands on the still moving UGV, and then the navigation as a whole finishes.

The formation errors during the navigation are shown in Fig. 9, where the starting points of the second and third laps are represented. Regarding \( \tilde{x} \) and \( \tilde{y} \), it is apparent that the error peaks are smaller at the first lap, slightly larger in the second lap and larger in the third lap. This happens because the sudden change of the desired path to avoid the obstacles increases the error momentarily. However, one can see that the controller is capable of reducing those errors and that the maximum magnitude is of such errors is in the order of 20 centimeters. The error in the \( z \)-axis, \( \tilde{z} \), is always zero.
FIGURE 9. Formation errors.

because the virtual robot correspondent to the reference for the formation, which coincides with the UGV, is considered to be navigating always on the $x - y$ plane, where $z = 0$.

Additionally, one can compare the maximum errors $\tilde{x}$ and $\tilde{y}$ with the error bounds present in [40] for a similar speed value. There, the error bound can get as high as 50 centimeters.

The second column in Fig. 9 depicts the errors regarding the shape of the formation. The error associated with the length of the 3D Line, $\tilde{\rho}$, is quickly reduced to a value close to zero, which corresponds to the composition of the desired formation, and then is kept under control. Accordingly, the error $\tilde{\beta}$ exhibits the same behaviour. The error $\tilde{\alpha}$, however, is much larger. This happens because when $\beta = \frac{\pi}{2}$, as it is the case in $q_{d,n}$, $\alpha$ is actually in a singularity point, and thus any value at all will accomplish the task. Because of this, the error $\tilde{\alpha}$ can be disregarded in this situation (a more thorough explanation about such a singularity and why it can be disregarded is given in [15]).

Finally, Fig. 10 compares the desired velocities of the UGV, which represents the virtual robot that is the reference for the formation, with the measured values. The first graph shows the actual linear velocity in continuous blue line and the output of the controller in dashed red line, while the dashed yellow line represents the desired velocity $V_d$ along the path. The results show that the tracking of the desired speed is close to perfection, since the three curves are very similar to one another. Even so, one can note slight delays whenever there is a sudden drop in the reference velocity, caused by the strategy adopted to make severe turns, discussed in Section II. This happens because of the lack of a dynamic compensator for the UGV, which would deal exactly with the imperfections in the velocity tracking, but even without such a compensator, as it is the case here, the delay is still small enough to be disregarded in this application.

Due to the nonholonomic restriction of the UGV, the angular velocity is not freely set, as the linear velocity is. It is a consequence of the direction of the latter one, since the robot must always face the direction of its movement. Because of this, there is no reference value for the angular velocity, but nonetheless it is vital that it is correctly controlled, otherwise the robot will not be able to follow the desired direction. Fortunately, it can be seen that the robot successfully tracked the controller references for its angular velocity. Thus, as the robot has successfully tracked both the desired linear velocity and angular velocities, it ensures that both the magnitude and direction of the desired velocity along the path being followed are correct.

V. CONCLUSION

This paper proposed an obstacle avoidance algorithm embedded in a controller designed to guide an heterogeneous formation of robots (a UAV and a UGV) in a path-following task. In order to do that, the formation was characterized and the path following controller was adapted to embed the obstacle avoidance algorithm. Still considering the proposed path-following controller, a strategy was applied to adjust the desired velocity along the path, which allows the user to set a maximum velocity, instead of a constant one, which may increase efficiency when completing the task. 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.

Afterwards, the obstacle avoidance algorithm was formalized, first considering just single obstacles, and then was extended to handle multiple obstacles at once. This is done through establishing safe zones around the obstacles, shaped as circles, and then navigating along temporary safe paths generated by the algorithm until reaching another point on the path located after the obstacles. In the case of single obstacles, the safe path is always the shortest arc of the circle delimiting the safe zone, as both circumference arcs are equally safe. On the other hand, for multiple obstacles the safe path will be the safest circumference arc, whether it is the shortest or the longest one, and it will be established in a case by case basis in real time.
From the results presented, involving both simulations and experiments, it is possible to claim that the controller is effective in guiding the formation along a path, and that, by utilizing the proposed obstacle avoidance algorithm, the formation is able to navigate safely in an environment with previously unknown hazards.

As future work, it is under study to modify the multiple obstacles algorithm to model the multiple obstacles as a single ellipsis, instead of modelling separate safe zones as circles. The idea is that this might increase efficiency in the navigation. Also, the approach used to bring the points of interest to a new frame centered on the obstacle already makes such modification easier, because while the circle is immune to rotation around its center, an ellipsis is not, so its definition, and that of the shortest and longest arcs, are simpler when it is located at the origin with no rotation. As a second part, it is being studied how to handle moving obstacles, and the ability of the path controller to freely change the formations velocity is a great advantage, where in certain cases the most natural approach may be to stop the robots and wait until the moving obstacles do not represent a problem for the formation, or to change the formations velocity in such a way that the trajectory of the formation and of the obstacles do not cross over one another at the same time.

REFERENCES

[1] D. K. Villa, A. S. Brandão, and M. Sarcinelli-Filho, “A survey on load transportation using multicopter UAVs,” J. Intell. Robot. Syst., vol. 98, pp. 267–296, May 2020.

[2] I. Palunko, P. Cruz, and R. Fierro, “Agile load transportation: Safe and efficient load manipulation with aerial robots,” IEEE Robot. Autom. Mag., vol. 19, no. 3, pp. 69–79, Sep. 2012.

[3] S. Dai, T. Lee, and D. S. Bernstein, “Adaptive control of a quadrotor UAV transporting a cable-suspended load with unknown mass,” in Proc. 53rd Ann. Conf. Decis. Control (CDC), Dec. 2014, pp. 6149–6154.

[4] C. Meissen, K. Klausen, M. Arcak, T. J. Fossen, and A. Packard, “Passivity-based formation control for UAVs with a suspended load,” IFAC-PapersOnline, vol. 50, no. 1, pp. 13150–13155, 2017.

[5] B. Shirani, M. Najafi, and I. Izadi, “Cooperative load transportation using multiple UAVs,” Aerosp. Sci. Technol., vol. 84, pp. 158–169, Jan. 2019.

[6] D. K. D. Villa, A. S. Brandao, and M. Sarcinelli-Filho, “Rod-shaped payload transportation using multiple quadrotors,” in Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS), Atlanta, GA, USA, Jun. 2019, pp. 1036–1040.

[7] I. H. B. Pizetta, A. S. Brandão, and M. Sarcinelli-Filho, “Load transportation by quadrotors in crowded workspaces,” IEEE Access, vol. 8, pp. 223941–223951, 2020.

[8] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, “Vehicle planning with multi-fidelity models for agile flights in unknown environments,” in Proc. Int. Conf. Robot. Autom. (ICRA), 2019, pp. 725–731.

[9] 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 Trans. Mechatronics, vol. 22, no. 6, pp. 2543–2553, Dec. 2017.

[10] B. Kovács, G. Szayer, F. Tajit, M. Burdelis, and P. Korondi, “A novel potential field method for path planning of mobile robots by adapting animal motion attributes,” Robot. Auton. Syst., vol. 82, pp. 24–34, Aug. 2016.

[11] T. A. Cerqueira, T. L. M. Santos, and A. G. S. Conceição, “A new approach based in potential fields with obstacles avoidance for mobile robots,” in Proc. Latin Amer. Robot. Symp. Brazilian Robot. Symp. (LARS/SBR), Oct. 2016, pp. 229–233.

[12] Y. C. Liu, S. Y. Liu, and N. Wang, “Fully-tuned fuzzy neural network based robust adaptive tracking control of unmanned underwater vehicle with thruster dynamics,” Neurocomputing, vol. 196, pp. 1–13, Jul. 2016.

[13] J. K. Pothal and D. R. Parhi, “Navigation of multiple mobile robots in a highly cluttered terrains using adaptive neuro-fuzzy inference system,” Robot. Auton. Syst., vol. 72, pp. 48–58, Oct. 2015.

[14] H. D. Mathias and V. R. Ragusa, “An empirical study of crossover and mass extinction in a genetic algorithm for pathfinding in a continuous environment,” in Proc IEEE Congr. Evol. Comput. (CEC), Jun. 2016, pp. 4111–4118.

[15] Z. Yan, J. Li, G. Zhang, and Y. Wu, “A real-time reaction obstacle avoidance algorithm for autonomous underwater vehicles in unknown environments,” Sensors, vol. 18, no. 2, p. 438, Feb. 2018.

[16] R. Wang, H. Jing, C. Hu, F. Yan, and N. Chen, “Robust \( H_\infty \) path following control for autonomous ground vehicles with delay and data dropout,” IEEE Trans. Intell. Transp. Syst., vol. 17, no. 7, pp. 2042–2050, Oct. 2016.

[17] Z. Zabierzewicz, “Robust and adaptive path-following control of an underactuated ship,” IEEE Access, vol. 8, pp. 120198–120207, 2020.

[18] J. Wang, C. Wang, Y. Wei, and C. Zhang, “Three-dimensional path following of an underactuated uav based on neuro-adaptive command filtered backstepping control,” IEEE Access, vol. 6, pp. 74355–74365, 2018.

[19] C. Shen, Y. Shi, and B. Buckham, “Path-following control of an UAV: A multiobjective model predictive control approach,” IEEE Trans. Control Syst. Technol., vol. 27, no. 3, pp. 1334–1342, May 2019.

[20] I. Palunko, P. Cruz, and R. Fierro, “Agile load transportation: Safe and efficient load manipulation with aerial robots,” in Proc. Int. Conf. Robot. Autom. (ICRA), May 2019, pp. 8041–8047.

[21] E. W. Dijkstra, “A note on two problems in connexion with graphs,” Numer. Math., vol. 1, no. 1, pp. 269–271, Dec. 1959.

[22] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” IEEE Trans. Syst. Sci. Cybern., vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[23] R. Bellman, “On a routing problem,” Quart. Appl. Math., vol. 16, no. 1, pp. 87–90, 1958.

[24] L. R. Ford, Jr., D. R. Fulkerson, and S. W. T. C. Chiang, “Networks,” Santa Monica, CA, USA: RAND Corp., 1956.

[25] K. K. Kwakniewski and Z. Giszewski, “Genetic algorithm for mobile robot route planning with obstacle avoidance,” Acta Mechan. Automat., vol. 12, no. 2, pp. 151–159, Jun. 2018.

[26] J. Tordesillas, B. T. Lopez, J. Carter, J. Ware, and J. P. How, “Real-time planning with multi-fidelity models for agile flights in unknown environments,” in Proc. Int. Conf. Robot. Autom. (ICRA), 2019, pp. 725–731.

[27] 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 Trans. Mechatronics, vol. 22, no. 6, pp. 2543–2553, Dec. 2017.

[28] T. A. Cerqueira, T. L. M. Santos, and A. G. S. Conceição, “A new approach based in potential fields with obstacles avoidance for mobile robots,” in Proc. Latin Amer. Robot. Symp. Brazilian Robot. Symp. (LARS/SBR), Oct. 2016, pp. 229–233.
V. P. Bacheti et al.: Path-Following Controller for UAV-UGV Formation Performing Final Step of Last-Mile-Delivery

[38] C. Z. Resende, R. Carelli, and M. Sarcinelli-Filho, “A path-following controller for guiding a single robot or a multi-robot system,” J. Control, Autom. Electr. Syst., vol. 32, no. 4, pp. 895–909, Aug. 2021.

[39] M. F. Reis, R. P. Jain, A. P. Aguiar, and J. B. de Sousa, “Robust cooperative moving path following control for marine robotic vehicles,” Frontiers Robot. AI, vol. 6, p. 121, Nov. 2019.

[40] D. Fridovich-Keil, S. L. Herbert, J. F. Fiscac, S. Deglurkar, and C. J. Tomlin, “Planning, fast and slow: A framework for adaptive real-time safe trajectory planning,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Oct. 2018, pp. 387–394.

[41] N. Mohanty, A. Singhal, T. K. Dan, and P. B. Sujit, “LQR-based cooperative path following of multiple UAVs with collision avoidance,” in Proc. 6th Indian Control Conf. (ICC), Dec. 2019, pp. 385–390.

[42] C. B. Low, “A flexible virtual structure formation keeping control design for nonholonomic mobile robots with low-level control systems, with experiments,” in Proc. IEEE Int. Symp. Intell. Control (ISIC), Oct. 2014, pp. 1576–1582.

[43] A. S. Brandao, V. T. L. Rampinelli, F. N. Martins, M. Sarcinelli-Filho, and R. Carelli, “The multilayer control scheme: A strategy to guide robots formations with obstacle avoidance,” J. Control, Autom. Electr. Syst., vol. 26, no. 3, pp. 201–214, Jun. 2015.

[44] 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,” J. Intell. Robot. Syst., vol. 93, nos. 1–2, pp. 5–16, Feb. 2019.

[45] 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 Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS), May 2014, pp. 756–767.

[46] A. O. Pinto, H. N. Marciano, V. P. Bacheti, M. S. M. Moreira, A. S. Brandao, and M. Sarcinelli-Filho, “High-level modeling and control of the bebop 2 micro aerial vehicle,” in Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS), Athens, Greece, Sep. 2020, pp. 939–947.

[47] H. K. Khalil, Nonlinear System, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

ALEXANDRE SANTOS BRANDÃO received the B.S. degree in electrical engineering from the Federal University of Viçosa, Minas Gerais, Brazil, in 2006, the M.Sc. and Ph.D. degrees in electrical engineering from the Federal University of Espírito Santo, Brazil, in 2008 and 2013, respectively, and the Ph.D. degree in control system engineering from the National University of San Juan, San Juan, Argentina, in 2014. He is currently an Associate Professor with the Department of Electrical Engineering, Federal University of Viçosa, Brazil. He has coauthored more than 20 journal articles, two book chapters, and more than 120 conference papers, besides having advised eight M.Sc. and four Ph.D. students. His research interests include nonlinear control of aerial and terrestrial mobile vehicles and control of multi-robot systems.

MÁRIO SARCINELLI-FILHO received the B.S. degree in electrical engineering from the Federal University of Espírito Santo, Brazil, in 1979, and the M.Sc. and Ph.D. degrees in electrical engineering from the Federal University of Rio de Janeiro, Brazil, in 1983 and 1990, respectively. He is currently a Professor with the Department of Electrical Engineering, Federal University of Espírito Santo, Brazil, and a Researcher with the Brazilian National Council for Scientific and Technological Development (CNPq). He has coauthored more than 60 journal articles, more than 340 conference papers, and 17 book chapters. He has also advised 20 Ph.D. students and 25 M.Sc. students. His research interests include nonlinear control, mobile robot navigation, coordinated control of mobile robots, unmanned aerial vehicles, and coordinated control of ground and aerial robots. He is also a Senior Editor of the Journal of Intelligent and Robotic Systems.

VINÍCIUS PACHECO BACHETI received the B.S. degree in electrical engineering from the Federal University of Viçosa, Brazil, in 2016, and the M.Sc. degree in electrical engineering from the Federal University of Espírito Santo, Brazil, in 2021, where he is currently pursuing the Ph.D. degree in electrical engineering. His research interest includes control of multi-robot systems applied to load transportation with aerial robots.