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

Integration of a 4D-trajectory follower to improve multi-UAV conflict management within the U-Space context [⋆]

Hector Perez-Leon · Jose Joaquin Acevedo · Ivan Maza · Anibal Ollero

Received: date / Accepted: date

Abstract A safe integration of UAVs into the airspace is fundamental to unblock all the potential of drone applications. U-space is the drone traffic management solution for Europe, intended to handle a large number of drones into the airspace, especially at Very Low Level (VLL). This paper is focused on conflict management for multiple unmanned aerial vehicles in the context of the U-space under 4D trajectory based operations (4D-TBO). A novel method for multi-UAV conflict management at tactical level for large-scale scenarios is presented. The integration of 4D-TBO in this context has been implemented with a four dimensional trajectory follower based on the carrot chasing algorithm. This method minimizes, through the whole flight, the mean normal distance to the defined trajectory and the mean difference with respect to the defined arrival times. Finally, the integrated system has been implemented in a software in the loop environment with a commercial autopilot. The simulation results show better performance with respect to other classical approaches.

Keywords Multi UAV · Conflict management · 4D-trajectory follower

1 Introduction and related work

It is envisioned that in the following decade "very-low-level" (VLL) air traffic will increase exponentially, mainly by the commercial use of drones. In order to maintain the level of risk in aerial operations, as in the last two decades, it will be necessary to develop a number of new services and specific procedures

[⋆] A version of this paper, entitled *A 4D trajectory follower based on the 'Carrot chasing' algorithm for UAS within the U-space context*, appeared in the Proceedings of The 2020 International Conference on Unmanned Aircraft Systems (ICUAS20), Athens, Greece.

Hector Perez-Leon
Robotics, Vision and Control Group
Avda de los Descubrimientos s/n, 41092, Seville, Spain
E-mail: hectorperez@us.es

that conforms the so-called U-space (drone traffic management solution for Europe). Due to the large number of drones that are expected to operate in the next years, these services will require a high level of digitalization and automation to facilitate not only a safe integration but also a secure and efficient one. Therefore, the development of U-space is crucial to boost the drone market and their public acceptance [1] in the following years. U-space is currently defined as a set of services [2] and procedures to support safe, efficient, and secure access to airspace for UAVs.

In this context, the concept of multi-UAV conflict management becomes a key challenge that is considered at two levels: strategic (pre-flight approaches) and tactical (in-flight approaches). On one hand, at strategic level most works focus on studying the probability of conflicts between pairs of UAVs, as in [3]. A quite relevant work, related to the strategic multi-trajectory conflict detection, is [4]. They propose the sequential division of the airspace to get sub-spaces containing an unique trajectory. However, its performance may decrease significantly for too dense scenarios. In [5], the authors incorporate intent information from predefined trajectories to complement a *Velocity Obstacle* approach, but focusing on the 2D problem. On the other hand, at tactical level conflict detection is also a well studied problem in the literature. The conditions to define a potential conflict between two UAVs, based on their tracked trajectories, are stated in [6]. Authors in [7] pose a hierarchical approach, which reduces the computation time to detect potential imminent conflicts by checking the relative distance between pairs of UAVs in an asynchronous way. In [8], they show how the intended flight plan may enhance the conflict detection in a tactical phase. Another reactive approach rather spread is using the concept of Velocity Obstacles [9]. These methods search a velocity space to assign the vehicles velocities leading to collision-free trajectories in future. Another geometric approach based on space-time prisms is proposed in [10].

Another related area is multi-vehicle path finding, which is known to be NP-hard optimization problem [11] where a set of vehicles need to find collision-free paths. Different optimization methods have been applied to solve the problem in continuous [12] or discrete space [13]. Centralized constrained optimization solvers have been proposed. For instance, [14] propose to model the problem as a task allocation, and then compute optimal trajectories in terms of traveled distance. In [15], linear programming in a velocity space is used. Also, Sequential Convex Programming has been proposed to obtain solutions in non-convex scenarios [16]. The main issue with these methods for multi-vehicle path finding is that they do not scale well with the size of the team.

Finally, the concept of 4D-trajectory Based Operations (4D-TBO) is another relevant issue in this context. It consists in the integration of the temporal dimension into the traditional flight plans, which included only the intended three dimensions spatial trajectory. Therefore, any delay in the time schedule should be assumed as a separation from the intended trajectory, just as a vertical or horizontal deviation. This concept is specially relevant for U-Space tactical services, such as monitoring and tactical deconfliction, where

4D-TBO increase the aerial traffic predictability, maximizing the airspace capacity and improving the overall safety in the aerial traffic management (less unexpected conflicts). Robust, efficient and precise autonomous following of predefined trajectories is a requirement by the UAVs for all these applications. The trajectory tracking problem for UAVs is well studied in the literature, and there are different geometric or control methods. Pure pursuit [17], carrot chasing [18], line-of-sight (LOS) [19] methods and vector field [20] are some common geometric algorithms.

This paper presents a method for multi-UAV conflict management at tactical level, based on the estimation of 4D UAV trajectories. The proposed solution represents the scenario by means of a 4D grid and uses a geometric approach to resolve conflicts in an iterative manner, minimizing the deviation with respect to the initial estimated trajectory. In short, our main contributions are the following: (i) we propose a novel method for multi-UAV conflict management at tactical level for large-scale scenarios within the U-space concept; and (ii) the integration of 4D-TBO in this context with a four dimensional trajectory follower based on the carrot chasing algorithm previously presented by the authors [21].

The rest of the paper is structured as follows. First, Section 2 states the conflict detection, tactical deconfliction and 4D-trajectory tracking problems. In Section 3, a conflict detection and resolution system and a 4D-trajectory follower are presented. Section 4, summarizes the set of simulation results used to validate the proposed approach. Finally, Section 5 closes the paper with the conclusions and future work.

2 Problem statement

We address conflict detection and resolution for multi-UAV systems from a tactical point of view and based on 4D-trajectories. There are several challenges and complementary issues to take into account to approach this problem.

2.1 Conflict definition

A conflict between two UAVs (so-called *loss of minimum separation* event, according to U-space terminology) may be defined as a situation where they approach below their *safety distances*. Depending on the type of vehicle, different separations parameters for the lateral, longitudinal and vertical dimensions may be considered and different shapes around the vehicle may be considered as their safety zones: a elliptical cylinder, an ellipsoid or a sphere.

Hereafter, we will consider multi-rotor UAVs, since they are the most usual drone type in very low level airspace. Thus, considering that the motion of a multirotor may be assimilated as holonomic under certain conditions, a spherical safety zone and a single safety distance parameter δ is applied, as it is shown in Figure 1. A conflict between two UAVs will happen at time t if and only if $|p_i(t) - p_j(t)| \leq \delta$, being $p_i(t)$ and $p_j(t)$ the positions at time t of UAVs A_i and A_j, respectively.

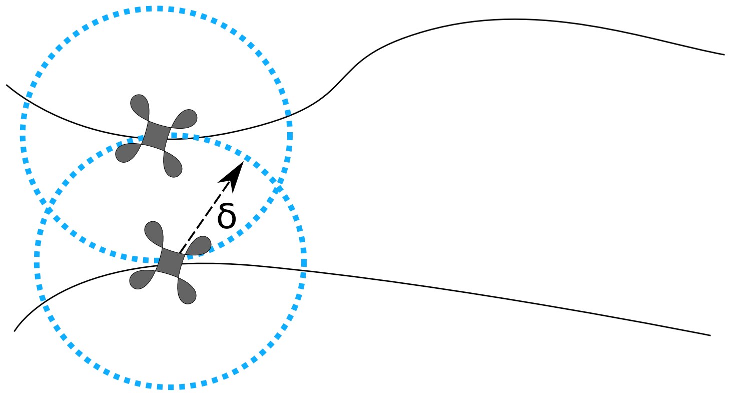


Fig. 1 Conflict situation between two multi-rotors. Gray lines indicate the trajectories of the UAVs and the dotted circles bounds up their safety zones. It is assumed that both UAVs are flying at the same altitude.

From a tactical point of view, the conflicts can not be detected at the current time t, but predicted with the enough time-span to decide and execute the most proper actions to avoid the potential conflict, i.e. following an alternative flight plan. Therefore, in order to handle the conflict detection problem it is required to have the estimated UAV poses along time for a given time horizon, i.e. the 4D-trajectories of the UAVs.

Let assume a 4D-trajectory as an ordered set of waypoints, being it defined by an expected 3D-position and its estimated arrival time. Let us also consider that every pair of consecutive waypoints are equally time-spaced by the same *inter-waypoint period* τ for any trajectory. Therefore, a potential conflict between a pair of trajectories will happen if there exist a pair of waypoints of both trajectories with a time difference below the inter-waypoint period and distance below the *safety distance*, see Figure 2.

2.2 Tactical deconfliction problem

Let us consider a set of 4D-trajectories for UAVs sharing a common airspace defined within a given time horizon. The conflict detection and resolution problem implies not only to detect potential conflicts according to the definitions of the trajectories presented in Section 2.1, but also to propose an alternative set of 4D-trajectories which ensure no potential conflicts among them.

There are infinite possible solutions to the posed problem. In order to assess relevant solutions, it will be assumed that the initial trajectories are the desired ones from an operational point of view, since they come from the flight plans provided by the UAV operators. Therefore, the mean distance deviation from

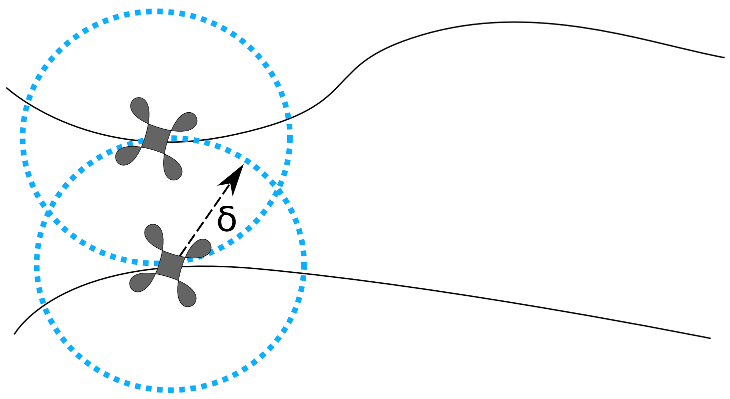


Fig. 2 Conflict between a pair of waypoints of two different trajectories. Black points indicate the waypoints. It is assumed that both trajectories have been coordinated to share the same altitude and longitudinal position. Therefore, the difference in the lateral position corresponds to the distance between them.

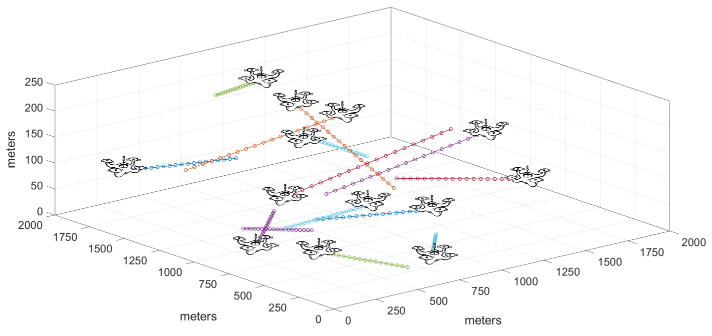


Fig. 3 Estimated trajectories within a given time horizon of several UAVs sharing a common airspace. Each waypoint is marked as a colored circle.

the initial trajectories to the alternative ones will be chosen as a minimization criteria. This distance deviation may be calculated as the 3D-distance between a pair of waypoints with the same estimated arrival time from the alternative and initial trajectories.

It is interesting to take into account the priorities associated to the different UAVs to solve the problem. A potential conflict between two UAVs with different priorities should be solved varying only the trajectory of the UAV with lower priority. Therefore, the optimization criteria should consider to weight the trajectory deviations according to these priorities.

On the other hand, since this problem should be solved in-flight and the alternative trajectories have to be submitted, accepted and executed by the UAVs, the required processing time to obtain a suitable solution has to be minimized.

2.3 4D-Trajectory tracking problem

The problem described in Section 2.2 is based on the accurate tracking of 4D-trajectories by the involved UAVs: each UAV has to follow its assigned waypoints matching each associated arrival time. Delays or advances with respect to the estimated arrival time may cause unexpected conflicts and requires to update the estimation of the trajectories, to detect new conflicts and to generate alternative trajectories continuously.

However, commercial autonomous navigation systems for UAVs do not usually manage 4D-trajectories, and sets a cruise flight speed to follow a given 3D-path. The 4D-trajectory could hardly be tracked using these systems, even when the arrival times associated to each waypoint had been properly chosen according to the UAV cruise speed. Moreover, matching the specified arrival times manually is not an easy task.

Let us assume that a UAV can be directly controlled through 3D-velocity commands. Known the intended 4D-trajectory of the UAV, the objective is to implement a system which tracks accurately the positions and the associated arrival times based on velocity commands. The maximum flight speed should be also considered for the UAV.

Formally, the criteria to minimize are two: the mean minimum distance between the actual travelled trajectory and the estimated one; and the mean difference between the actual and the estimated arrival times to every waypoint.

3 Solution adopted

The proposed solution is based on the assumption that the updated versions of the estimated 4D-trajectories for all the UAVs which share the controlled airspace are continuously available within a given time horizon T. These trajectories are defined as a ordered set of 4D-waypoints, equally time-spaced by τ seconds.

Conflict detection and resolution software is executed on a central ground station, which receives information from all the UAVs. It detects potential loss of separation events between UAVs and provides alternative plans to avoid them. On the other hand, for each UAV, its on-board computer executes a trajectory following algorithm. It receives the alternative plans provided by the ground segment and commands the UAV to match the 4D-trajectories as close as possible (not only in the space, but also in time). Figure 4 shows this software architecture.

3.1 Conflict detection and resolution based on 4D-grids

It is based on two interconnected services: monitoring and tactical deconfliction. Monitoring is in charge of detecting potential loss of separation events

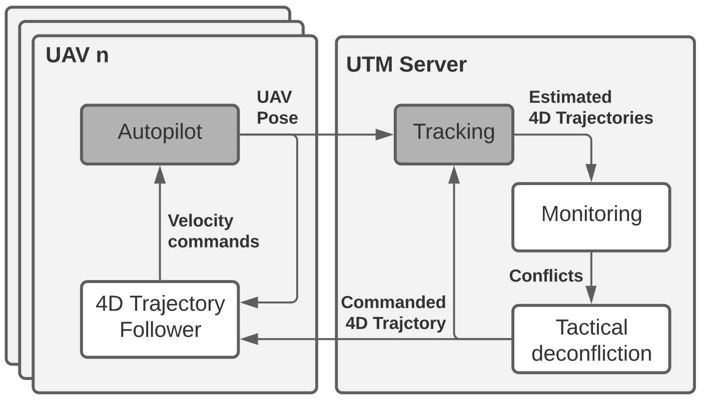


Fig. 4 Interactions between air (UAV) and ground (UTM) software modules. Shaded boxes are out of the scope of this paper.

and reporting to the tactical deconfliction algorithm which computes alternative trajectories for the involved UAVs. In turn, the alternative trajectories generated by the deconfliction service will be executed by the UAVs and registered by the monitoring service. Therefore, if the new trajectories cause new conflicts, they will be reported again from the monitoring to the deconfliction node, generating an iterative process.

3.1.1 Monitoring service

It is based on modeling the considered scenario as a 4D-grid. The whole controlled airspace within a time horizon T is divided into 4D-cells of size $dX \times dY \times dZ \times \tau$, being τ the length associated to the time dimension, see Figure 5. Each cell stores a list with the waypoints from the received set of 4D-trajectories which are into its associated 4D-space.

Periodically (each τ seconds), the monitoring service updates the 4D-grid using the estimated trajectories of the UAVs. Conflict evaluation is made only between waypoints which are in the same or neighboring cells. Therefore, when an updated waypoint is stored into its associated cell, the neighboring cells are checked to find potential conflicts, see Figure 6. This approach reduces dramatically the total amount of required checks with respect to other methods based on an exhaustive search approach, specially for large-scale scenarios.

After each iteration, the monitoring service provides a list of potential conflicts between pairs of waypoints of different trajectories, including their information.

Since the grid has to be stored in the computer, its size is directly related to the memory storage requirements and algorithm efficiency. Let us consider a controlled airspace of size $dX \times dY \times dZ$ and a time horizon of T seconds. Then, the dimensions of the required grid would be $\lceil X/dX \rceil \times \lceil Y/dY \rceil \times \lceil Z/dZ \rceil \times \lceil T/\tau \rceil$, being $\lceil \rceil$ defined as the ceiling function.

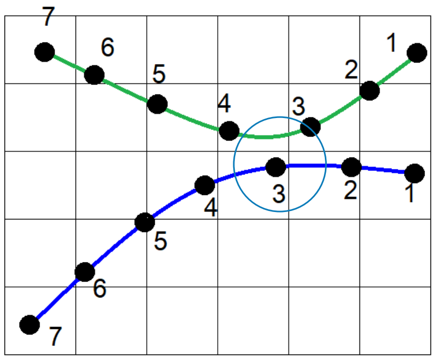


Fig. 5 Airspace shared by two trajectories divided into a grid. The altitude is not shown for the sake of clarity. Black circles represent the waypoints and the associated number its arrival time.

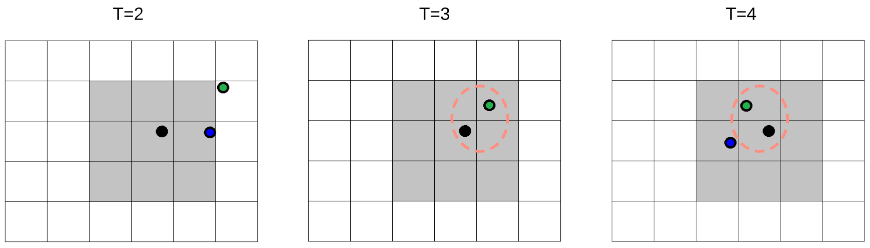


Fig. 6 Conflict detection process associated to the third waypoint from the blue trajectory in the scenario represented in Figure 5. This waypoint is stored into its cell and its neighboring cells (dashed cells in the figure) are checked to look for waypoints from other trajectories (green trajectory, in this case). The analyzed waypoint has to be validated against the third and four waypoint from the green trajectory, as it is shown in the example.

Another issue to consider is the lower limit of the cell size. On one hand, since conflicts are checked only between neighboring cells, the size of the cells cannot be less than the safety distance δ. In other case, a potential conflict could exist between two waypoints which are not in the neighboring cells. Therefore, $dX, dY, dZ \geq \delta$.

On the other hand, the cell size has to be bounded down by the speed of the involved UAVs. The UAVs cannot fly fast enough to go through two cells in a single period τ because potential loss of separation events could be ignored. Therefore, defining $v_{\max}$ as the maximum speed of the fastest UAV involved in the scenario, the size of the cell cannot be less than $\tau v_{\max}$: $dX, dY, dZ \geq \tau v_{\max}$.

The relation between the safety distance and the 4D-cell dimensions influences the system efficiency. On one hand, the lower the value of the safety distance, the bigger the 4D-grid size, memory requirements and time to create the grid, but lower processing time since the number of checks between waypoints in neighboring cells will be minimized. On the other hand, as larger these dimensions are, the conflict detection algorithm will look more like an exhaustive search.

3.1.2 Tactical deconfliction

This algorithm is based on a geometric approach to solve sequentially each conflict between pairs of waypoints. The idea is separating each pair of conflicting waypoints independently and taking advantage of the periodic checks performed by the monitoring service to solve iteratively more complex situations (with more than two UAVs involved).

For each conflict received, it computes a new pair of waypoints separated enough to match the safety distance, as it is shown in Figure 7. These new waypoints are calculated modifying the 3D-position, but not the arrival time, generating an estimated variation of the nominal speed of the UAV with respect to the original one. Then, they are fused with the associated trajectories and reported to the corresponding UAVs.

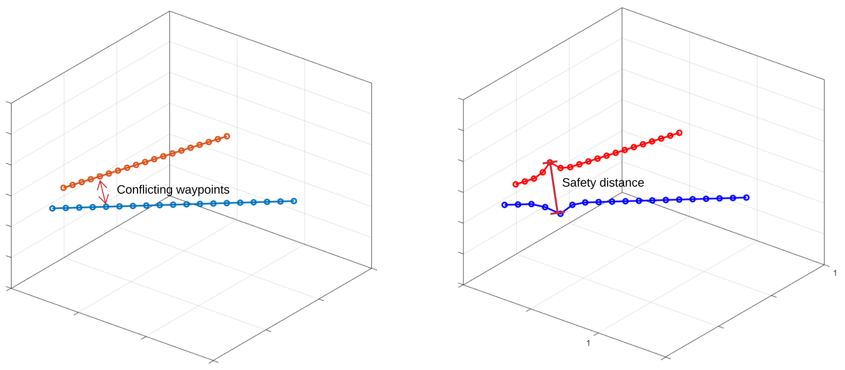


Fig. 7 On the left two trajectories with a pair of waypoints in conflict. On the right, the alternative trajectories provided by separating the conflicting waypoint to match the safety distance.

In order to calculate the new waypoints, different issues have to be considered. First, UAVs operations may be assigned with different priorities. We keep without variation the trajectories associated to UAVs with higher priorities, proposing alternative trajectories for the UAVs with lower priorities.

In addition, waypoints separation may be performed following different directions: the direction which join both waypoints, vertically or horizontally. The first option seems to be the more efficient from an airspace capacity point of view and according to the conditions specified in Section 2.2. However, the latter options are safer since they are more predictable for the rest of manned and unmanned aircrafts.

It should be noticed that the proposed approach requires that UAVs can adapt their flight velocities to properly follow the generated trajectories. Although initial flight plans could have been chosen to fly at a single nominal speed, alternative trajectories do not respect this principle.

Although each conflict is solved independently, i.e. not considering the rest of the trajectories, the monitoring service runs periodically and generates an iterative process which is illustrated in Figure 8.

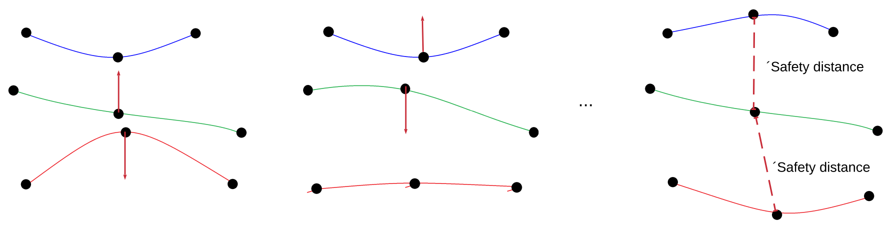


Fig. 8 Separation between three trajectories (waypoints) based on several iterations (from left to right).

Finally, it has to be checked if the UAVs can follow the alternative trajectory, matching the arrival time requirements according to its speed capabilities. In summary, this one may be assured always the required cruise speed of the UAV to perform the initial plan is lower than the half of its maximum speed. In case this condition is not met, different strategies may be adopted: modifying also the previous and/or next waypoint to the conflicting one; or assigning a different separation weight to each conflicting waypoint depending on the difference between the original cruise speed and the maximum speed.

3.2 4D-trajectory follower based on the 'Carrot chasing' algorithm

It endows the generator and the follower algorithms as it is shown in Figure 9.

The generator algorithm creates a much more dense list of waypoints based on the ordered list of waypoints received and it can be approximated to a continuous curve. To follow a 4D-trajectory, it should interpolate the initial list of times matching the amount of waypoints of the more dense list. The follower algorithm minimizes the distance between the actual travelled trajectory and the estimated one; and the mean difference between the actual and estimated arrival times to the waypoints. It receives the generated 4D-trajectory, the look ahead distance, the maximum speed and, continuously, the UAV pose and the actual time. The follower algorithm calculates the velocity command as follows: First, the point on the trajectory that matches the minimum distance between the UAV and the trajectory should be obtained. The look ahead

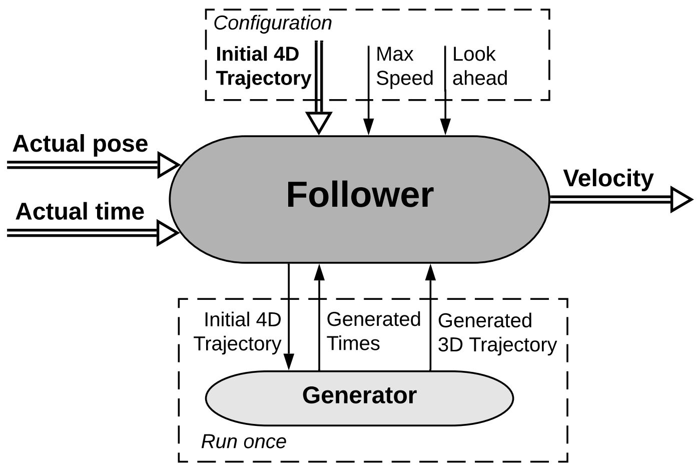


Fig. 9 The trajectory follower design allows to use it by simply configuring the initial 4D waypoint list. It also provides more configuration options to suit the user needs. The generator is called by the follower and runs once to generate a discrete curve with the desired time on each point of the curve.

distance is added to get the virtual target pose on the trajectory. To fix the actual time error, the method should calculate the cruising speed subtracting the actual time to the arrival time of the target waypoint and dividing it by the look ahead distance. If the UAV is ahead of schedule, the cruising speed will be reduced, and otherwise the method will increase the cruising speed to fix the actual time error. The last step is to obtain the unitary vector between the UAV pose and the virtual target pose and multiply it by the cruising speed previously calculated. Figure 10 shows a graphical representation of the method.

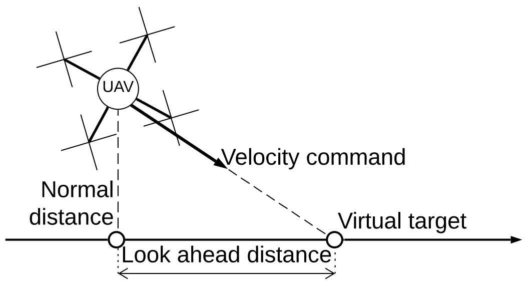


Fig. 10 Top view of the four dimensional trajectory follower based on the carrot chasing algorithm without taking into account the orientation error.

4 Validation results

This section details the different simulation tests carried out in order to validate the approaches proposed in this paper. Large scale simulations based on MATLAB are provided to analyze the scalability of the conflict detection and resolution approach and ROS-Gazebo based tests are performed to validate the whole system integration running in-flight.

4.1 Scalability analysis

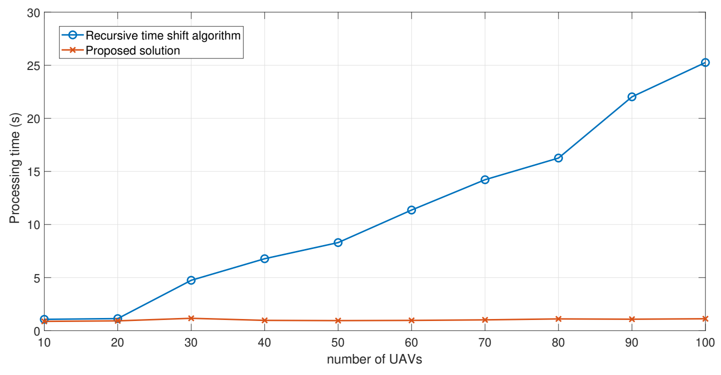
More than 500 MATLAB simulations for large-scale scenarios have been carried out on an Intel i7@2.2GHz (RAM 8 GB) CPU, in order to analyse the scalability of the developed conflict detection and resolution system. For this analysis, no priorities are considered. Let us consider a controlled airspace volume of dimensions $5000 \times 5000 \times 200$ meters and a safety distance of 50 meters between each pair of UAVs. It is assumed that the monitoring service receives the updated estimated trajectories of the UAVs as ordered lists of 4D-waypoints equally time-spaced by 5 seconds. Assuming a maximum flight speed limited to 20 m/s for all the UAVs, the initial UAV trajectories are randomly generated following a straight line and matching the speed requirements defined in Section 3.1.2 (the half of the maximum speed, it means 10 m/s).

Time-based algorithms are a traditional solution for air traffic management conflicts. Thus the proposed solution is compared against a conflict resolution approach based on the recursive time shift algorithm [22]. It proposes to shift the trajectories in time, starting at the conflicting waypoints, to find the successful trajectories. Regarding to the conflict detection approach, a detailed comparison with respect to the traditional exhaustive search approach was presented in [23].

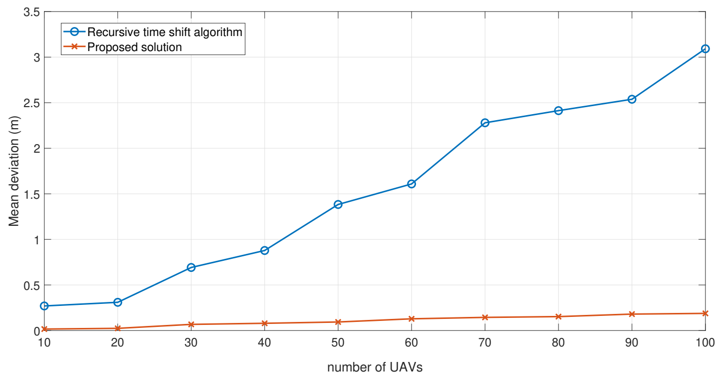
A first element which can influence the optimization criteria described in Section 2.2 is the number of involved UAVs. Therefore, a test battery increasing the number of UAVs and setting the time horizon to 100 seconds and the 4D-cell dimensions to $100 \times 100 \times 100$ meters was carried out. Figure 11 shows the processing time and the mean deviation with respect to the original trajectories using the proposed and the alternative approaches.

The time horizon is another interesting element whose impact in the system efficiency should be analyzed. A large time horizon may be necessary in order to match the U-space requirements, allowing the UAV operators to receive, analyze and execute the proper commands to avoid the potential conflicts. Therefore, a second test battery increasing the time horizon and setting the number of involved UAVs to 30 and the 4D-cell dimensions to $100 \times 100 \times 100$ meters has been performed. Processing time and mean deviation with respect to the original trajectories are shown in Figure 12.

The results show that the proposed solution requires much less processing time and gets much less deviation with respect to the original trajectories than



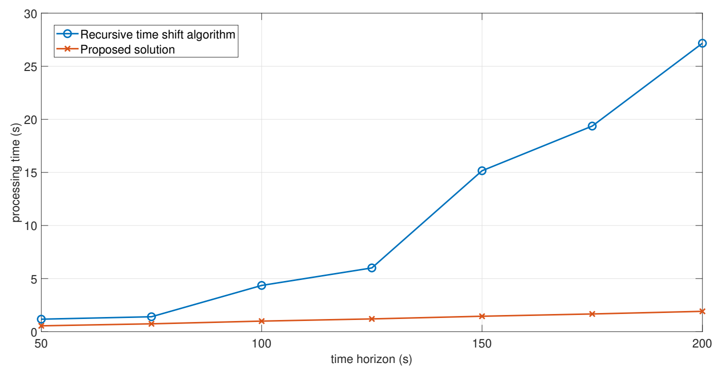
(a)



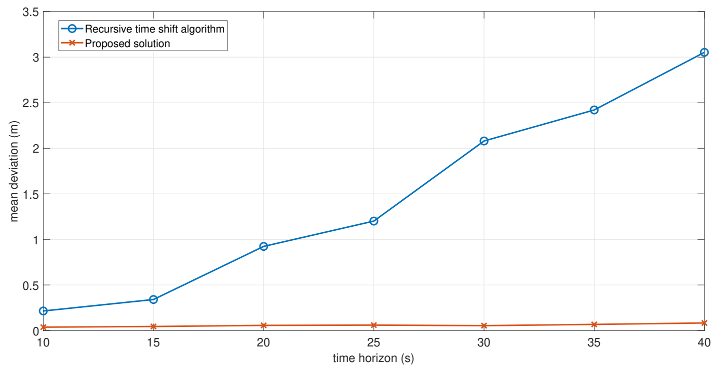
(b)

Fig. 11 Summary of the test battery increasing the number of UAVs and comparing the proposed solution against the time shift based approach: (a) Mean processing time by test to detect and resolve all the conflicts. (b) Mean deviation by waypoint between the generated and the original trajectories.

the more traditional recursive time shift algorithm. These improvements are highly related to the number of involved trajectories and the considered time horizon. The proposed solution exploits the assumption of tracking accurately the trajectory both in time and space to modify only the conflicting waypoint. However, shifting a trajectory in time implies to modify every waypoint in the trajectory, generating new conflicts which have to be detected and solved again.



(a)



(b)

Fig. 12 Summary of the test battery increasing the time horizon and comparing the proposed solution against the time shift based approach: (a) Mean processing time by test to detect and resolve all the conflicts. (b) Mean deviation by waypoint between the generated and the original trajectories.

4.2 In-flight tests

The proposed solution was tested on the *Robotic Operating System* [1] (ROS) framework [24] (Kinetic distribution) under the Ubuntu 16.04 operating system using GAZEBO [25], the PX4 v1.7.3 *Software In The Loop* (SITL) [26] functionality to simulate the autopilot and the UAV Abstraction Layer (UAL) v3.0 [27] to interact with the simulated UAVs. This simulation environment allows to run and try different tests without flying a real UAV using the same software. The previous section presented a scalability analysis running test involving 100 UAVs, this section is focused on the behaviour of UAVs on the simulation framework, thus the in-flight tests use 3 UAVs to clarify the results.

[1] https://www.ros.org/

The following tests use multiple UAVs with flight plans that have the same characteristics, such as the longitude (100 meters), the arrival time of the last waypoint (37.5 seconds) and the cruising speed (2.7 m/s), but they differ in the origin and arrival poses. The conflict detection and resolution system has a safety distance δ of 10 meters and an inter-waypoint distance τ of 5 seconds. For each multi-UAV test, two plots are provided: 3D visualization of the UAVs travelling along their flight plans and the distance between the UAVs involved in the test.

Figure 13 shows an example of these flight plans in detail, with a trajectory that can be followed for the simulated UAV which has a maximum velocity of 4 m/s. Figure 13a has different scales on axes to better visualize the error between trajectories. Figure 13b shows how the trajectory follower cap the commanded velocity to the maximum velocity in the first seconds. Figure 13d shows the difference between the UAV actual time and the desired time: if it is negative the UAV is behind of schedule, otherwise the UAV is ahead of schedule. The 4D-trajectory follower always tries to make the time difference equal to zero. It also tries to minimize the normal distance between the UAV and the trajectory. The space and time errors that the 4D-trajectory follower tries to minimize are shown in Table 1.

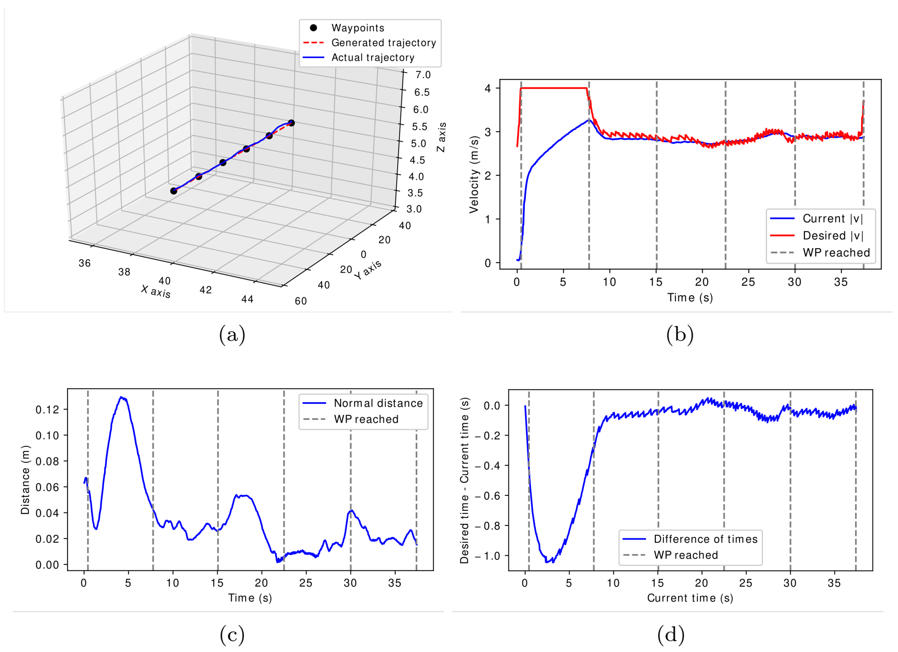
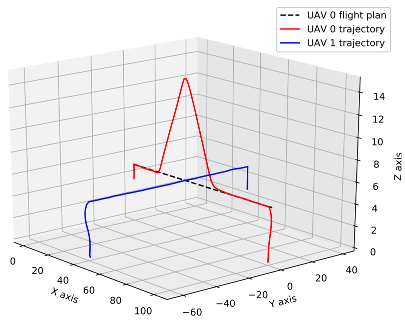


Fig. 13 Simulation results. (a) Three dimensional view of the initial waypoints, the reference trajectory generated and the actual trajectory flown. (b) Desired velocity and current velocity of the UAV. (c) Normal distance between the UAV and the generated trajectory. (d) Difference between the desired time and the current time.

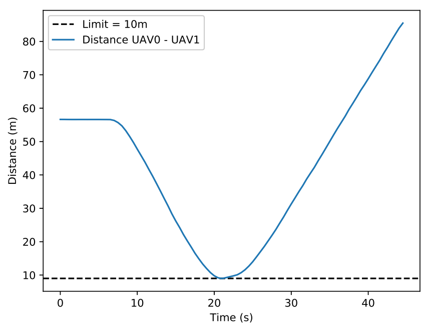
Table 1 Simulated test errors

	Min	Mean	Max	Std	Var
Space Error (m)	0.002	0.036	0.129	0.030	0.001
Time Error (s)	0.000	0.196	1.047	0.311	0.097

The first in-flight test presents a conflict after 15 seconds between two UAVs that start their flight plans at the same time. UAVs 0 and 1 have priorities 0 and 1 respectively. The conflict detection and resolution module decides to leave one flight plan as it is and modifies the other flight plan due to the UAVs priorities to solve the conflict. Despite the increment of the flight plan longitude, the UAV manages to fly through the trajectory in time because it uses the 4D-trajectory follower, which minimizes its time and space errors. UAV 0 finishes its flight in 37.46 seconds even modifying its flight plan and UAV 1 finishes in 37.51 seconds. Figure 14b shows the distance between UAVs which never goes below the limit which is 10 meters. A 3D visualization of the trajectories can be seen in Figure 14a.



(a)

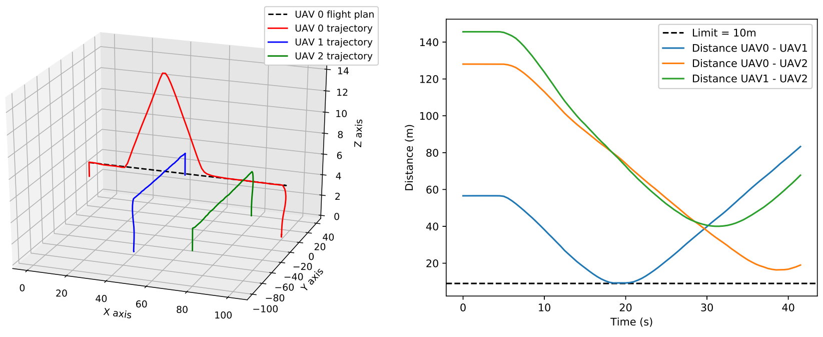


(b)

Fig. 14 (a) 3D Visualization of the traveled trajectories. (b) Distances between UAVs. UAV 0 has a conflict with UAV 1 but the solution of the conflict detection and resolution module keeps the distance above the limit.

The second in-flight test presents three UAVs flying with flight plans similar to the ones used in the previous simulations except for UAV 2 which has a flight plan that lasts 7.5 seconds more. The conflict between UAV 1 and UAV 2 is found in the same position and time that happened in the previous test. As priorities do not change, the conflict detection and resolution module calculates the same solution. The solution of the conflict between UAV 0 and UAV 1 does not create another conflict between UAVs 0 and 2, and they fly through the same waypoint at different times so the minimum distance between UAVs is not violated (UAV 0 went through the waypoint at 30 seconds and UAV 2 at

37.5 seconds). Figure 15b shows the distances between the UAVs which never goes below the limit. UAV 0 finishes its flight in 38.14 seconds even modifying its flight plan, UAV 1 finishes in 37.44 seconds and UAV 2 in 44.94 seconds. A 3D visualization of the trajectories can be seen in Figure 15a and a video of the simulation is available at https://youtu.be/0U42krj1MTM.



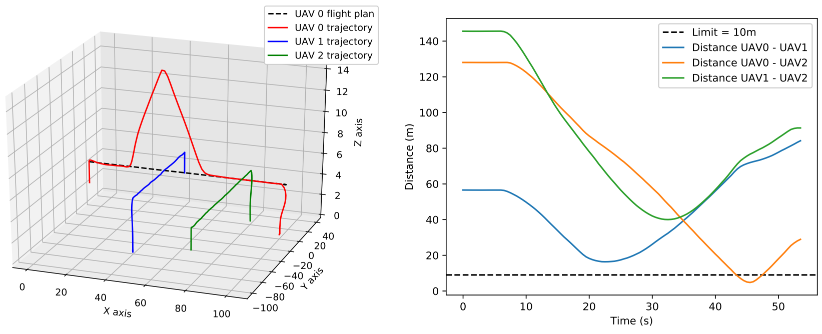
(a) (b)

Fig. 15 (a) 3D Visualization of the traveled trajectories. (b) Distances between UAVs. UAV 0 has a conflict with UAV 1 but the solution of the conflict detection and resolution module keeps the distance above the limit. UAV 0 is using the 4D-trajectory follower.

The third in-flight test presents the relevance of using a 4D-trajectory follower. The conflict detection and resolution module gives its solution to the UAV 0 which follows the new trajectory using a cruising speed of 2.7 m/s to match the mean of the velocity of the last test. The new trajectory is longer than the initial flight plan, so while going along the modified trajectory to avoid UAV 1, UAV 0 is delayed. This causes a conflict between UAVs 0 and 2 where their flight plans cross at 37.5 seconds which did not appear in the previous test and the distance between UAVs 0 and 2 goes below the limit, see Figure 16b. The new trajectory causes a new conflict, it will be reported again from the monitoring to the deconfliction node, generating an iterative process to solve the conflict, but in this in-flight test, the conflict will not be solved to emphasize that the distance between UAVs goes below the limit. A 3D visualization of the trajectories can be seen in Figure 16a and a video of the simulation is available at https://youtu.be/8oKwk7tL-dI.

5 Conclusions and future work

The use of UAVs will be increased in the next decades, being the air traffic management much more complex. In this context the concept of 4D-trajectory becomes a key element to increase the flight safety and to maximize the shared airspace usage. This paper faces this challenge, presenting a conflict detection



(a) (b)

Fig. 16 (a) 3D Visualization of the traveled trajectories. (b) Distances between UAVs. UAV 0 has a conflict with UAV 1 but the solution of the conflict detection and resolution module keeps the distance above the limit. UAV 0 is not using the 4D-trajectory follower, and it causes a conflict with UAV 2, lowering their distance below the limit.

and resolution module for multiple UAVs and a 4D-trajectory follower with low deviation in distance and time.

The proposed approach to detect and solve loss of separation events takes advantage of the assumption that UAVs may track accurately the trajectories in time and space. Based on this, the resolution approach only modifies the conflicting waypoints positions (not their arrival times), requiring the velocity adaptation from the UAV. The results are significantly better than other accepted approaches as the recursive time shift approach with respect to the required processing time and the mean deviation from the original trajectory.

The developed UAV 4D-trajectory follower can fix the space and time errors while tracking a given trajectory. It minimizes at every time the mean minimum distance between the actual travelled trajectory and the estimated one, and the mean difference between actual and estimated arrival times to every waypoint.

Future developments will be directed to test the conflict management system for long distance missions in outdoors experiments using different commercial GNSS receivers to validate the scalability of the solution. In addition, the proposed system will be integrated in a framework for the development and testing of UTM functionalities, including pre-flight and in-flight U-Space services.

Acknowledgements *This work is partially supported by the GAUSS (H2020-GALILEO-GSA-776293) European project.

References

1. M. Macias, C. Barrado, E. Pastor, P. Royo, in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)* (IEEE, 2019). DOI 10.1109/dasc43569.2019.9081623. URL https://doi.org/10.11092Fdasc43569.2019.9081623
2. CORUS Consortium, U-space concept of operations. Tech. rep., European Commission (2019)
3. W. Liu, I. Hwang, Journal of Guidance, Control, and Dynamics **34**(6), 1779 (2011). DOI 10.2514/1.53645. URL https://doi.org/10.2514/1.53645
4. A. Kuenz, N. Peinecke, in *2009 IEEE/AIAA 28th Digital Avionics Systems Conference* (2009), pp. 3.B.5–1–3.B.5–10. DOI 10.1109/DASC.2009.5347522
5. G.A. Mercado Velasco, C. Borst, J. Ellerbroek, M.M. van Paassen, M. Mulder, IEEE Transactions on Intelligent Transportation Systems **16**(4), 2297 (2015). DOI 10.1109/TITS.2014.2376031
6. A. Alonso-Ayuso, L.F. Escudero, P. Olaso, C. Pizarro, TOP **21**(3), 485 (2013). DOI 10.1007/s11750-011-0224-6. URL https://doi.org/10.1007/s11750-011-0224-6
7. J. Yang, D. Yin, Y. Niu, L. Zhu, in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (2015), pp. 2436–2441. DOI 10.1109/ROBIO.2015.7419704
8. H. Tang, J. Robinson, D. Denery, in *10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference* (American Institute of Aeronautics and Astronautics, 2010). DOI 10.2514/6.2010-9294
9. E. Lalish, K.A. Morgansen, Autonomous Robots **32**(3), 207 (2012). DOI 10.1007/s10514-011-9267-7. URL http://dx.doi.org/10.1007/s10514-011-9267-7
10. H. Siqi, S. Cheng, Y. Zhang, Chinese Journal of Aeronautics **31**(7), 1579 (2018)
11. W. Honig, S. Kiesel, A. Tinka, J.W. Durham, N. Ayanian, IEEE Robotics and Automation Letters **4**(2), 1125 (2019). DOI 10.1109/lra.2019.2894217
12. D. Mellinger, A. Kushleyev, V. Kumar, 2012 IEEE International Conference on Robotics and Automation pp. 477–483 (2012). DOI 10.1109/ICRA.2012.6225009
13. J. Yu, S.M. Lavalle, IEEE Transactions on Robotics **32**(5), 1163 (2016). DOI 10.1109/TRO.2016.2593448
14. M. Turpin, K. Mohta, N. Michael, V. Kumar, Autonomous Robots **37**(4), 401 (2014). DOI 10.1007/s10514-014-9412-1
15. I. Karamouzas, S.J. Guy, 2015 IEEE International Conference on Robotics and Automation (ICRA) pp. 5983–5989 (2015). DOI 10.1109/ICRA.2015.7140038
16. J. Alonso-mora, E. Montijano, M. Schwager, D. Rus, 2016 IEEE International Conference on Robotics and Automation (ICRA) pp. 5356–5363 (2016). DOI 10.1109/ICRA.2016.7487747
17. R.C. Coulter, Implementation of the pure pursuit path tracking algorithm. Tech. rep., Carnegie-Mellon UNIV Pittsburgh PA Robotics INST (1992)
18. A. Micaelli, C. Samson, Trajectory tracking for unicycle-type and two-steering-wheels mobile robots. Ph.D. thesis, INRIA (1993)
19. T.I. Fossen, M. Breivik, R. Skjetne, IFAC Proceedings Volumes **36**(21), 211 (2003)
20. D.R. Nelson, D.B. Barber, T.W. McLain, R.W. Beard, IEEE Transactions on Robotics **23**(3), 519 (2007)
21. H. Perez-Leon, J.J. Acevedo, I. Maza, A. Ollero, in *Proc. of the International Conference on Unmanned Aircraft System (ICUAS)* (2020)
22. N. Peinecke, A. Kuenz, AIAA/IEEE Digital Avionics Systems Conference - Proceedings **2017-Septe** (2017). DOI 10.1109/DASC.2017.8102048
23. J.J. Acevedo, .R. Castao, J.L. Andrade-Pineda, A. Ollero, in *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)* (2019), pp. 18–23. DOI 10.1109/REDUAS47371.2019.8999724
24. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A.Y. Ng, ICRA workshop on open source system (2009)
25. N. Koenig, A. Howard, in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3 (2004), vol. 3, pp. 2149–2154 vol.3. DOI 10.1109/IROS.2004.1389727
26. L. Meier, D. Honegger, M. Pollefeys, in *Proceedings of the IEEE International Conference on Robotics and Automation* (2015). DOI 10.1109/ICRA.2015.7140074

27. F. Real, A. Torres-González, P. Ramón-Soria, J. Capitán, A. Ollero, International Journal of Advanced Robotic Systems **17**(4), 172988142092501 (2020). DOI 10.1177/1729881420925011