# Providing resilience to UAV swarms following planned missions

Jamie Wubben[1], Izan Catalán[1], Manel Lurbe[1], Francisco Fabra[1],
Francisco J. Martinez[2], Carlos T. Calafate[1], Juan-Carlos Cano[1], Pietro Manzoni[1]
[1]*Departament of Computer Engineering (DISCA)*
*Universitat Politècnica de València, Valencia, Spain*
[2] *iNiT Research Group, Computer Science and System Engineering Department*
*University of Zaragoza, Spain*
Email: jwubben@disca.upv.es, {izcagal,malursem}@inf.upv.es, frafabco@cam.upv.es,
f.martinez@unizar.es, {calafate,jucano,pmanzoni}@disca.upv.es

***Abstract*—As we experience an unprecedented growth in the field of Unmanned Aerial Vehicles (UAVs), more and more applications keep arising due to the combination of low cost and flexibility provided by these flying devices, especially those of the multirrotor type. Within this field, solutions where several UAVs team-up to create a swarm are gaining momentum as they enable to perform more sophisticated tasks, or accelerate task execution compared to the single-UAV alternative. However, advanced solutions based on UAV swarms still lack significant advancements and validation in real environments to facilitate their adoption and deployment. In this paper we take a step ahead in this direction by proposing a solution that improves the resilience of swarm flights, focusing on handling the loss of the swarm leader, which is typically the most critical condition to be faced. Experiments using our UAV emulation tool (ArduSim) evidence the correctness of the protocol under adverse circumstances, and highlight that swarm members are able to seamlessly switch to an alternative leader when necessary, introducing a negligible delay in the process in most cases, while keeping this delay within a few seconds even in worst-case conditions.**

***Index Terms*—UAV, swarm, resilience, ArduSim, flight coordination.**

## I. Introduction

Unmanned Aerial Vehicles (UAVs) are currently considered a hot topic, as evidenced by the many works on this subject, as well as the attention from the media on a daily basis. In fact, their versatility, especially that of multirrotor UAVs, enables performing a wide range of tasks that were unthinkable just a few decades ago [1], like inspections at dangerous locations (e.g. nuclear plants), fire propagation analysis, rescue missions, or even traffic monitoring.

While currently most of these applications rely on the deployment of a single UAV, interest is growing in solutions where multiple UAVs are simultaneously deployed to perform a joint task [2], thereby conforming a UAV swarm. Examples of such applications include large-scale agriculture in search of pests or weeds [3], wild life recordings [4], or border surveillance [5], among others.

While technological advances have facilitated the control of multirrotor UAVs even for inexperienced users without a technological background, the situation differs drastically when a single user attempts to manage a UAV swarm. In fact, such option is not readily available in commercial devices, and only a few R&D teams worldwide have tried to face this challenge. Among the many issues that raise when handling a UAV swarm we could cite the takeoff and landing procedures [6], that should be fast while providing guarantees that UAVs shall not collide in the process, the election of swarm leader, the assignment of UAVs to specific positions in the swarm, and the flight coordination among UAVs [7].

While a coordinated flight is taking place, the issue of fault resilience also arises. We refer to a fault as some event that causes one or more UAVs to fail, thereby being removed from the swarm. Such condition can be particularly problematic when the swarm leader itself suffers the failure, which could potentially affect the entire swarm, impeding all active UAVs to continue their mission.

In this paper we specifically focus on the swarm resilience problem. In particular, we will show how a swarm protocol called MUSCOP [7] can be enhanced so as to be robust to the loss of the leader, or even to the loss of several UAVs, including leader and backup leader elements. By implementing our solution in a realistic UAV emulation platform, we are able to show the effectiveness of the proposed approach, and assess the time overhead introduced in the process.

The remainder of this paper is organized as follows: in section II we provide an overview of related works on this topic. Then, in section III, we present the MUSCOP protocol for UAV swarms following planned missions, and also our proposal to enhance this protocol in order to make it resilient to the loss of UAVs, especially the swarm leader. Next, section IV provides an overview of the simulation framework and the target metrics. Simulation results are then presented and discussed in section V. Finally, section VI presents the main conclusions of this paper, and refers to future works.

## II. Related works

Although not too many authors have addressed the specific topic of UAV swarms, it is possible to find some interesting works in the literature. For instance, Leonard et al. [8] have tested different swarm algorithms using quadcopters. They focused on a surveillance system for tracking selected targets.

978-1-7281-6607-0/20/$31.00 ©2020 IEEE

In their work, the swarm was modelled as a multi-agent system, and they sought an optimal cooperation among agents to maximize mission safety. Their solution was able to achieve smooth and safe navigation, even in dense environments.

Anwar Ma'sum et al. [9] developed a solution where a swarm of UAVs was able to perform object localization and tracking. The swarm robots were equipped with a Modified Particle Swarm Optimization (PSO) Algorithm that outperformed a fully random based moving algorithm for object localization and tracking. However, experiments were performed in a confined indoor environment of only $48m^2$. Furthermore, the object they were localizing had an orange color, a color that is easier to detect since it is not a common color in nature. So, confusing the object with other objects is less likely.

Wallar et al. [10] developed a path planning solution based on the concept of a scalable dynamic grid, where quadcopters cooperated to cover an environment in an efficient and reliable manner. Their approach registered already surveyed areas to avoid repeating analysis for those areas. In addition, the flight altitude of each UAV was determined using a nonlinear optimization. Their approach was tested in a simulated environment.

Pestana et al. [11] presented a modular multi-robot swarm architecture, where each swarm agent consists of an AR Drone 2.0 quadrotor connected to a laptop which runs the software architecture. Their approach relies on the Robot Operating System (ROS) software framework. This makes their work available for a great audience, since ROS makes code sharing and module reuse easy, and the AR Drone 2.0 is readily available on the market. In their approach, the only information shared among swarm agents is the position of each robot, and they rely on a visual-based solution for localization based on ArUco markers, which are used to sense and map obstacles. In addition, they rely on an Extended Kalman Filter localization and mapping method. However, their approach heavily depends on the WiFi links between the UAV and the laptop, which caused some problems according to the authors.

Mammen et al. [12] present an approach to designing swarms of autonomous, adaptive robots based on an observer/controller framework that has been developed as part of the Organic Computing initiative. Relying on an extended Learning Classifier System (XCS), in combination with adequate simulation techniques, it empowers the individuals to improve their collaborative performance, and to adapt to changing goals and changing conditions.

Sadrollah et al. [13] propose a distributed localisation framework for fast and reliable dissemination of localised information in elastic three-dimensional networks composed of UAV swarms. In particular, they combined IoT technologies with swarm robotics to control swarms. However, this work only introduces the basic idea, and no actual experiment was performed.

Mulgaonkar et al. [14] tested the performance of micro-quadcopter swarms in tight/dense formations, delta leader-follower and square formation flight experiments. The drones were also demonstrated to be robust to collisions at velocities of 4m/s. While we acknowledge the advantages of micro UAVs, we also believe that, due to their low mass, they are usually unable to withstand the elements in an outdoor environment. This makes the small, inexpensive and agile micro UAVs only useful for indoor applications, whereas we tend to focus on outdoor applications.

More recently, Bai et al. [15] proposed an improved UAV swarm model by incorporating the effect of a limited communications range. An improved resilience metric is proposed based on the difference between the swarm's performance and its standard system performance.

Our work significantly differs from the former ones as we specifically focus on mission-based swarm solutions, aiming at improving robustness to the loss of the swarm leader, a topic not addressed by any of the previous proposals.

## III. Proposed approach

In this section we detail our proposed solution, which improves upon an existing protocol for UAV swarm management that we have developed in a previous work, called MUSCOP [7], endowing it with resilience to the loss of swarm elements, especially focusing on the loss of the leader and backup leader UAVs. We will first provide a quick overview of MUSCOP, and then introduce the proposed improvements to make this protocol more robust.

### *A. MUSCOP overview*

The MUSCOP protocol aims at keeping a stable flight formation while the swarm follows a previously planned mission. It is based on the master-slave model, where the swarm is synchronized by the master UAV when swarm members reach each waypoint of the mission. Before taking off, all the slaves receive from the master a copy of the master mission, having the different waypoint coordinates adjusted so as to account for their position offset in the flight formation with regard to the swarm leader. During flight, UAVs move from waypoint to waypoint of their own mission, waiting on each of these waypoints until the master UAV starts the next part of the mission. As the planned speed is the same for all the UAVs in the swarm, this solution keeps the flight formation steady throughout the entire flight. For more details on the functionality and performance of this protocol, please refer to [7].

### *B. Resilience mechanisms*

The MUSCOP protocol described above has shown to operate consistently both in simulation and real environments, introducing a minimal overhead on each mission waypoint. However, its original design is not prepared to handle the loss of swarm elements. Hence, in order to make MUSCOP resilient to the loss of UAVs, especially the swarm leader, we have devised a set of mechanisms to make it more robust. The improvements made include modifications to the decisions and message exchanges taking place before the takeoff, and also during the flight itself. The code for this protocol is freely available online at [16].

*1) Message exchange:* The first aspect to take into account is that all the messages are transmitted continuously until the UAVs react accordingly. This way, we allow the flight to continue unhindered even when several messages are lost in transmission due to an unreliable communications link.

Figure 1 shows the strategy used by MUSCOP to coordinate the flight, while changing the leader of the swarm when needed. At the beginning, the leader is predefined by the user, and the slaves continuously send a *hello* message to inform about their presence, so that the leader can determine the size of the swarm. At the same time, it provides feedback to the pilot through the ArduSim graphical user interface, informing about the number of currently detected slaves. Once the pilot considers that all the UAVs in the swarm have been detected, he can start the setup step, where the leader calculates the corresponding location of each UAV in the flight formation, given their current location on the ground. When it finishes these calculations, it sends a *data* message to each slave including the mission to follow, and an ordered list of UAVs defining which one must become the leader during the flight in case the current leader fails (further explained in section III-B2). Regarding the mission each UAV must follow, it is a modified copy of the mission of the leader, considering the offset in the flight formation between each slave and the leader.

Once the master sends the *data* message to the slaves and it is acknowledged, it sends the *readyToFly* message to synchronize the response of the UAVs to the messages received. Only when all the slaves acknowledge the reception of this message does the master stop sending it, and performs the takeoff. Similarly, slaves will take off when a short timeout elapses since the last time they received the *readyToFly* message. The UAVs take off one by one using the takeoff sequence provided by the master UAV, and finally they inform the pilot that they are ready to start their mission.

*2) Determining the leader:* In order to create a list of leaders, we reuse an algorithm that we developed earlier called safeTakeOff [6]. We encourage the reader to refer to this paper to get a full understanding of its working principles. Nevertheless, we will provide a quick overview below.

The objective of the algorithm is to compute the assignment of UAV positions in a swarm. In order to do this efficiently it is necessary to minimize the overall distance travelled by each UAV during the takeoff process. In practical terms, this means that the UAVs moving to the most remote positions will take off first because this is safer and, it will also reduce the overall time overhead. This process is presented in Algorithm 1. Basically, it consist of the following four steps:

1) Find a central location with respect to the UAVs deployed on the ground,
2) Calculate the euclidean distances from that central location to the positions in the flight formation,
3) Sort this list in descending order,
4) Assign each location in the flight formation to the closest UAV on the ground, in descending order given that distance.

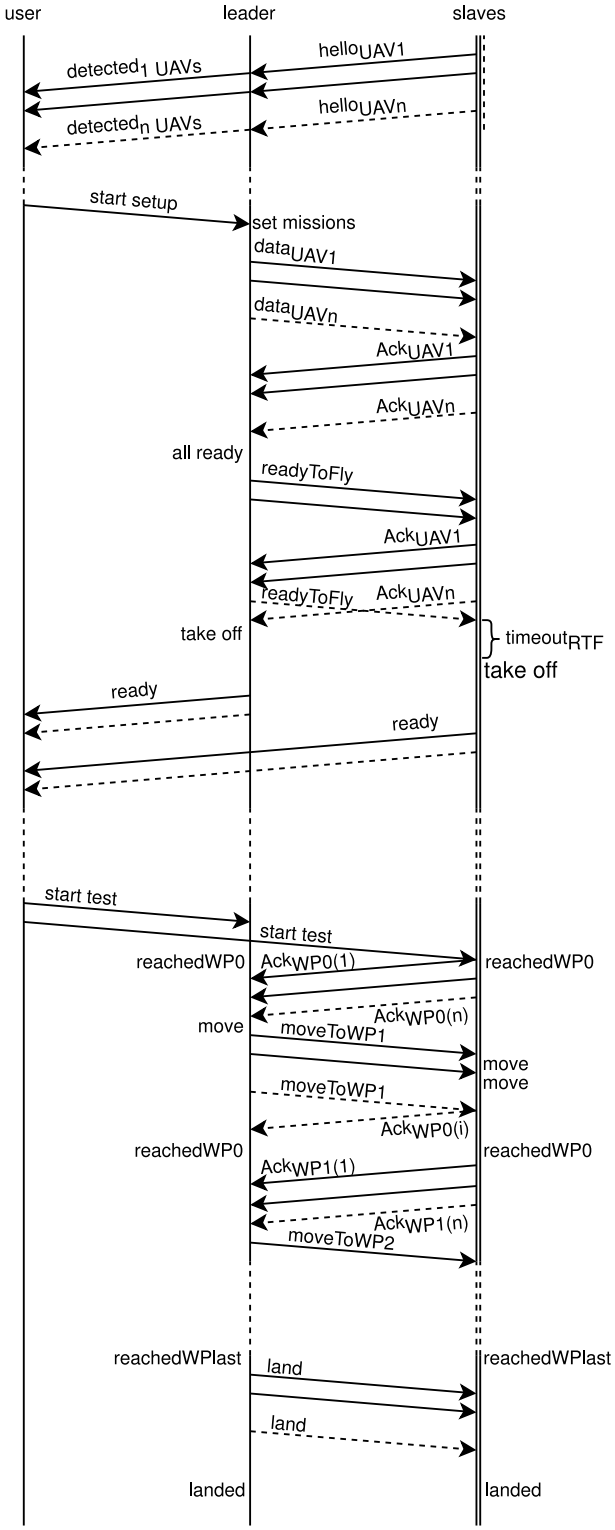


Fig. 1. Interaction between operator, master UAV and slaves.

To obtain the list including the leader and the backup leaders, we only have to reverse the takeoff sequence. We use this approach because the messages are sent from leader to slaves, and vice-versa, and this way we minimize the distance between sender and receiver, hence optimizing communications.

**Algorithm 1** SafeTakeOff(numUAVs, groundLocations, flightFormation)

**Require:** $groundLocations.size = numUAVs \wedge$ $flightFormation.size = numUAVs$

```
centerLocation = mean(groundLocations)
airLocations = f(centerLocation, flightFormation)
airList = ∅
for loc in airLocations do
    airList ← (loc, loc.distance(centerLocation))
end for
sort airList in descending distance order
fit = ∅
totalError = MAX_VALUE
for aLocation in airList do
    bestError = MAX_VALUE
    for gLocation in groundLocations do
        error = gLocation.distance(aLocation)^2
        if error < bestError then
            bestError = error
            bestID = gLocation.ID
        end if
    end for
    totalError += bestError
    fit ← (id, groundLocations[bestID], aLocation)
    groundLocations.remove(bestID)
end for
return fit
```

*3) Flight sequence:* The flight starts when the pilot uses the corresponding *start* command in ArduSim, sending a message to all the UAVs. The location in the flight formation is considered as the first waypoint of the mission, so all the slaves inform the leader that they have reached that waypoint. When the leader receives those acknowledgments, it starts sending the *moveToWP* command to force them to move to the next waypoint. The leader keeps sending this message even when it reaches the next waypoint in order to allow the slaves to keep track of its presence. The slave UAVs also start moving to the next waypoint when they receive the *moveToWP* command, and they also keep informing the leader that they have reached the previous waypoint until they arrive to the next one for the sake of redundancy. This process continues until all the UAVs reach the last waypoint. Then, the leader starts the landing procedure, sending them a message to force the slaves to land. Before landing, the slaves come nearer to the leader while maintaining the same flight formation in order to reduce the landing area, thus making it easier to collect the UAVs, avoiding them to land in far and/or inappropriate areas.

*4) Loss of leader:* In case the leader of the swarm fails while moving from one waypoint to the next, it stops sending messages to the slaves, and they detect the failure when the time elapsed since the last received message surpasses a certain threshold, which in our implementation was set to five seconds. The UAVs take the decision to change the leader of the swarm when they reach a waypoint and detect that the current leader has been lost. Then, they automatically switch the leader to the next one in the list provided by the master UAV, which details the order in which the UAVs must became leaders. The decision is taken in a distributed fashion, as each slave takes this decision on its own, and it requires a limited amount of time, as the slaves are typically able to detect that the leader has been lost prior to reaching the new location. If the following UAV in the list also fails, then the remaining UAVs switch to the next leader in the list, and the process goes on until no UAVs remain in the swarm. This way, the mission can be resumed even when more than one UAVs fails.

## IV. Simulation Framework

Our proposed solution was developed and evaluated using ArduSim, a novel multi-UAV flight simulator/emulator we have developed [17], and that is freely available online [18] under the Apache License 2.0. ArduSim is able to emulate the physics of hundreds of multicopters with great accuracy, being that software agents communicate with Ardupilot firmware through the MAVLink communications protocol [19]. It also simulates the communication between UAVs through virtual Wi-Fi links (ad hoc mode).

Figure 2 shows the main window of ArduSim. Most of the window area (1) shows the movement of the virtual multicopters, and the planned mission itself (if it applies). In this example, three UAVs are following a planned mission. Several buttons up on the right (2) allow to control the simulation, and help to show relevant data about each UAV in real time. Finally, the log in the upper left corner (3) shows the progress of the simulation, and messages generated by the protocol under development.

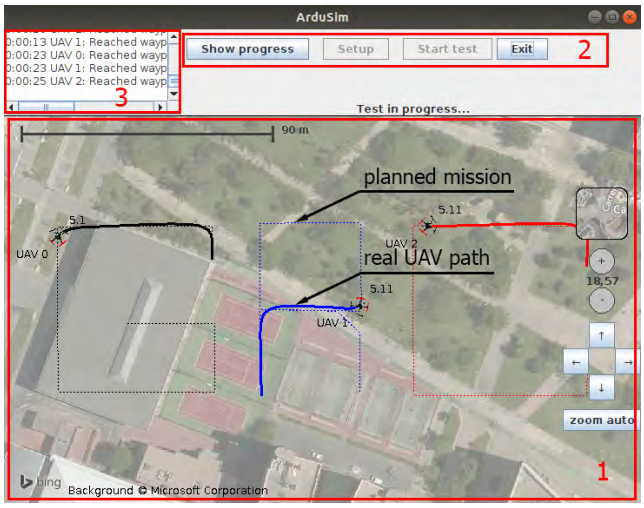


Fig. 2. Three UAVs following independent missions on ArduSim.

The most relevant characteristics of ArduSim are:

- Seamless protocol deployment on real UAVs.
- Soft real-time simulation.

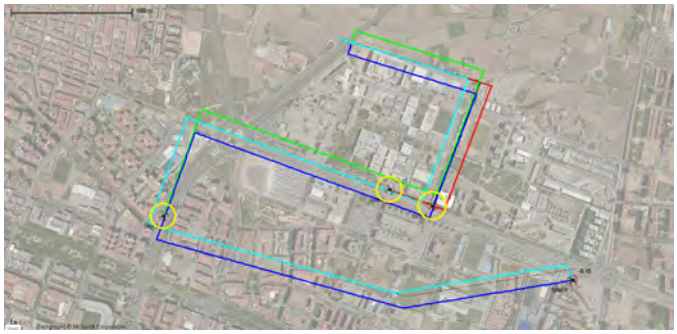

Fig. 3. Experiment where a swarm of 5 UAVs start a mission, but only 2 elements are able to complete it. Circles highlight UAVs that failed.

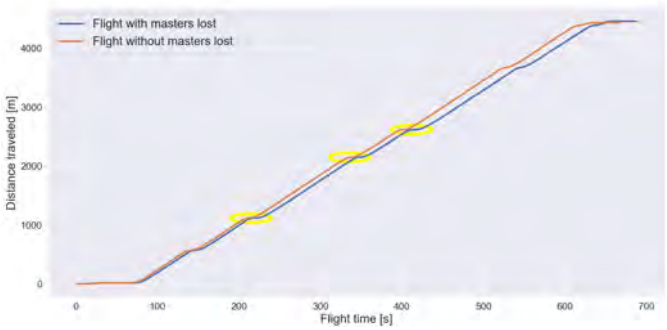


Fig. 4. Mission time comparing the default case (no UAV lost) against the loss scenario.

- Seamless scaling to more than 100 UAVs in real time.
- Wireless channel model for UAV-to-UAV communications.
- Complete Application Programming Interface (API).
- Automatic UAV collision detection.
- Comprehensive experiment data logging.

For the current study we have developed our proposed solution in ArduSim, so as to quickly evaluate its performance under different conditions. Specifically, we have forced the loss of leader and even a variable number of backup leaders in our experiments, so as to assess the time overhead involved in reconfiguring the swarm by picking a new leader among the available UAVs. Thus, our main performance metric was the time overhead per loss event, where a loss event may involve losing one or more UAVs.

## V. Simulation Results

Using the simulation framework described in section IV, we performed several experiments to validate the correctness of our solution, and to measure the responsiveness of the swarm management protocol to UAV loss events. The settings and results are summarized in Table I

TABLE I
SETTINGS AND RESULTS OF THE EXPERIMENTS PERFORMED

| | |
|---|---|
| Number of simulated UAVs | 5 |
| Number of lost leaders | 3 |
| Average time to detect leader loss | 2.6 s |
| Total flight time | 693 s |
| Overhead caused by protocol | 22 s |

In Figure 3 we show a snapshot of a simulation experiment where a swarm of 5 UAVs takes off and starts following a set of waypoints defined for the mission. Along the way different UAVs fail and land. In all cases the remaining UAVs continue their original mission unhindered. In this example, only 2 of the UAVs are actually able to complete the whole mission.

Figure 4 shows the simulation time differences corresponding to the experiment shown above, comparing it to the scenario without any loss of UAVs. The different steps shown correspond to the short pauses made by swarm elements at each waypoint. As can be observed, the overall delay at the end of the mission is of about 22 seconds, which represents a 3.3% increase in the overall flight time. In general we consider that this value can be considered acceptable for most application scenarios.

To gain more insight into the delays associated to the procedure of selecting an alternative leader for the swarm, Figure 5 shows the delay introduced in worst-case situations, that is, when the leader and backup leaders are lost, and when this event takes place just before reaching the next waypoint, thus leaving little reaction time for the remaining UAVs to pick a new leader. As shown, the highest value in this example is obtained when the leader and the next 4 backup leaders in the list are lost just when arriving to a waypoint; in this situation the delay is of about 9 seconds, 5 of which correspond to the timeout required to detect the current leader is lost, and 4 additional seconds are required to detect the other 4 backup leaders are lost as well. In typical cases, where the UAVs are lost in-between waypoints ($t \geq 5$ seconds), the delays are significantly lower, being of only 202 ms if the leader is lost, but the first backup leader is available.

## VI. Conclusions and Future Work

As UAV applications continue to grow in number and complexity, it becomes more and more important to achieve their goal in a safe and reliable manner. In this paper we

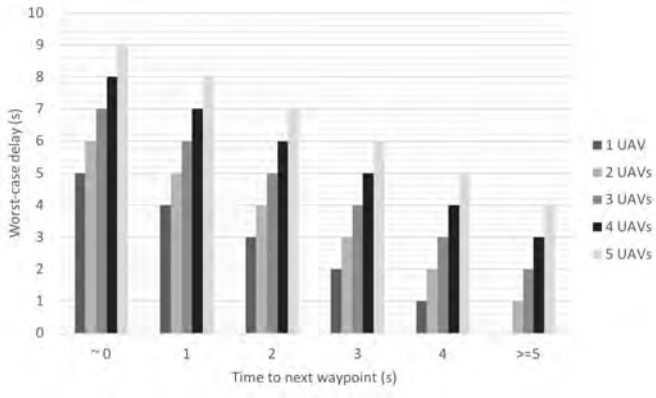


Fig. 5. Histogram showing the expected delay for the swarm when a variable number of UAVs is lost just before reaching the next waypoint.

have proposed a solution to provide resilience mechanisms to a swarm protocol. Our solution is able to efficiently handle the loss of the swarm leader, and even the loss of several UAVs including both leader and backup leaders. Experimental results using our ArduSim emulation tool show that the proposed approach is able to cope with the loss of swarm elements in an effective manner, while introducing a negligible delay to flight times in most of the cases. In worst-case scenarios, the delay can grow slightly to a few seconds per waypoint following a loss event, which is still a very reasonable value.

As future work we plan to validate the proposed solution in a more exhaustive manner, and address swarm split-up situations.

## ACKNOWLEDGMENT

This work was partially supported by the "Ministerio de Ciencia, Innovación y Universidades, Programa Estatal de Investigación, Desarrollo e Innovación Orientada a los Retos de la Sociedad, Proyectos I+D+I 2018", Spain, under Grant RTI2018-096384-B-I00.

## REFERENCES

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

[2] A. Tahir, J. Böling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila, "Swarms of unmanned aerial vehicles — a survey," *Journal of Industrial Information Integration*, vol. 16, pp. 1–7, 2019.

[3] D. Albani, J. IJsselmuiden, R. Haken, and V. Trianni, "Monitoring and mapping with robot swarms for agricultural applications," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug 2017, pp. 1–6.

[4] K. Anderson and K. J. Gaston, "Lightweight unmanned aerial vehicles will revolutionize spatial ecology," *Frontiers in Ecology and the Environment*, vol. 11, no. 3, pp. 138–146, 2013.

[5] E. Commission, "Autonomous swarm of heterogeneous RObots for BORDER surveillance," https://cordis.europa.eu/project/rcn/209949/factsheet/en, accessed: 2020-03-09.

[6] F. Fabra, J. Wubben, C. Calafate, J. Cano, and P. Manzoni, "Efficient and coordinated vertical takeoff of UAV swarms," in *IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, May 2020.

[7] F. Fabra, W. Zamora, P. Reyes, C. T. Calafate, J. Cano, P. Manzoni, and E. Hernandez-Orallo, "An UAV Swarm Coordination Protocol Supporting Planned Missions," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, July 2019, pp. 1–9.

[8] J. Leonard, A. Savvaris, and A. Tsourdos, "Towards a fully autonomous swarm of unmanned aerial vehicles," in *Proceedings of 2012 UKACC International Conference on Control*, Sep. 2012, pp. 286–291.

[9] M. A. Ma'sum, G. Jati, M. K. Arrofi, A. Wibowo, P. Mursanto, and W. Jatmiko, "Autonomous quadcopter swarm robots for object localization and tracking," in *MHS2013*, Nov 2013, pp. 1–6.

[10] A. Wallar, E. Plaku, and D. A. Sofge, "A planner for autonomous risk-sensitive coverage (parcov) by a team of unmanned aerial vehicles," in *2014 IEEE Symposium on Swarm Intelligence*, Dec 2014, pp. 1–7.

[11] J. Pestana, J. L. Sanchez-Lopez, P. de la Puente, A. Carrio, and P. Campoy, "A vision-based quadrotor swarm for the participation in the 2013 international micro air vehicle competition," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2014, pp. 617–622.

[12] S. von Mammen, S. Tomforde, J. Höhner, P. Lehner, L. Förschner, A. Hiemer, M. Nicola, and P. Blickling, "Ocbotics: An organic computing approach to collaborative robotic swarms," in *2014 IEEE Symposium on Swarm Intelligence*, Dec 2014, pp. 1–8.

[13] G. P. Sadrollah, J. C. Barca, A. I. Khan, J. Eliasson, and I. Senthooran, "A distributed framework for supporting 3d swarming applications," in *2014 International Conference on Computer and Information Sciences (ICCOINS)*, June 2014, pp. 1–5.

[14] Y. Mulgaonkar, G. Cross, and V. Kumar, "Design of small, safe and robust quadrotor swarms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 2208–2215.

[15] G. Bai, Y. Li, Y. Fang, Y.-A. Zhang, and J. Tao, "Network approach for resilience evaluation of a UAV swarm by considering communication limits," *Reliability Engineering & System Safety*, vol. 193, p. 106602, 2020.

[16] "ArduSim MUSCOP Protocol extension," https://github.com/mlurbe97/ArduSim-MUSCOP-Protocol-SRM, accessed: 2020-03-16.

[17] F. Fabra, C. T. Calafate, J.-C. Cano, and P. Manzoni, "ArduSim: Accurate and real-time multicopter simulation," *Simulation Modelling Practice and Theory*, vol. 87, no. 1, pp. 170–190, sep 2018. [Online]. Available: https://doi.org/10.1016/j.simpat.2018.06.009

[18] "ArduSim. accurate and real-time multi-UAV simulation," https://bitbucket.org/frafabco/ardusim/src/master/, accessed: 2020-03-09.

[19] L. Meier and QGroundControl, "MAVLink Micro Air Vehicle Communication Protocol," http://qgroundcontrol.org/mavlink/start, accessed: 2019-01-30.