Toward secure, efficient, and seamless reconfiguration of UAV swarm formations

Jamie Wubben[1], Pablo Aznar[1], Francisco Fabra [1],
Carlos T. Calafate[1], Juan-Carlos Cano[1], Pietro Manzoni[1]
[1]*Departament of Computer Engineering (DISCA)*
Universitat Politècnica de València, Valencia, Spain
Email: jwubben@disca.upv.es, pabazcol@alumni.upv.es, frafabco@cam.upv.es,
calafate@disca.upv.es, jucano@disca.upv.es, pmanzoni@disca.upv.es

***Abstract*—Unmanned Aerial vehicles (UAVs) have gained a lot of interested over the last years due to the many fields of potential application. Nowadays, researchers are becoming interested in groups of UAVs working together. The collaborations between UAVs open a wide field of opportunities, because they are typically able to do more sophisticated tasks than a single UAV. However, collaboration between multiple UAVs is still a complex task, and significant challenges need to be addressed before their mainstream adoption. For instance, the automatic reconfiguration of a swarm can be used to adapt the swarm to changing application demands to solve a task in a more efficient and effective manner. However, the chances of collision become high if reconfiguration is not carefully planned. In this work we propose an approach to allow changing the shape of a UAV formation during flight through a computational inexpensive method that is able to decrease collision chances significantly. During the experiments we tested different reconfiguration events that are prone to collisions. Results have shown that our approach significantly decreases the chances of collision while keeping the reconfiguration time overhead within reasonable bounds.**

I. Introduction

Over the last decade the field of Unmanned Aerial Vehicles (UAVs) has gained universal interest and novel applications keep emerging every year. Due to the ever decreasing price of technology, UAVs (also known as drones) are becoming mainstream for the general public and industry as well. This results in many civilian applications in aerial photography and video, topography, entertainment, etc. [1]. More professional applications such as precision agriculture, border surveillance, package delivery, and thermal inspections are also common in the industry [2], [3]. Nowadays, UAVs are starting to be used to assist in emergency situations such as search and rescue, or disaster scenarios [4], [5], where they can act as supporting nodes for communications being deployed on demand, and offering a wider communications range and better line-of-sight (LOS) features than ground infrastructures.

Over the last few years, the research works shifted more towards groups of coordinated UAVs [6]. Multi-UAV applications have great benefits as they are generally able to perform more sophisticated tasks efficiently or with more redundancy. However, organizing a multi-UAV flight is not an easy task, with challenges in terms of (i) swarm formation definition, (ii) takeoff procedure, (iii) in-flight coordination, (iv) swarm layout reconfiguration, (v) handling the loss of swarms elements, (vi) communications and data relaying optimization, and (vii) controlled landing, among others. In this work we focus on the particular problem of swarm reconfiguration during a mission. Notice that the ability to automatic change the shape of a formation during a mission can become very useful in different kinds of applications to account for: variable application requirements, coping with the loss of swarm elements, handling temporary flight restrictions, etc. For instance, consider a search and rescue mission where at first a swarm has to cover a large area but, upon discovering the item of interest, the swarm needs to reconfigure to better monitor that area and provide different services.

The main issue that we face during a reconfiguration is the chances of collisions, especially when the number of UAVs becomes larger. In this work we focus on a computational inexpensive technique to reduce the chances of collision that can be deployed easily under various conditions. Our solution combines two algorithms, the first determines the optimal assignment of UAVs in the new formation accounting for their current position, while the second one splits the UAVs in different mobility groups that are shifted to different altitudes during the reconfiguration process to minimize collision risks. Experimental results show that our solution is able to minimize collisions risks compared to other alternatives, while introducing only a moderate reconfiguration delay.

The rest of this paper is organized as follows: in Section II we provide an overview of related works on this topic. In Section III we detail our implementation. This implementation is then tested through different experiments, which are presented and discussed in Section IV. This work finishes with a critical discussion and the obtained conclusions in Section V.

II. Related work

The research towards swarms of UAVs has experienced a growing interest in recent years. The particular topic of flight configurations has been investigated by different authors. The work by V.T. Hoang et al. [7] presents an algorithm to reconfigure a formation of multiple UAVs. This work is especially focused on the application of vision-based inspection of infrastructure. It presents a new algorithm for reconfiguration based on the angle-encoded Particle Swarm Optimization (PSO). They begin with a 3D representation of

the surface to be inspected and a set of intermediate waypoints. Additionally new constraints are proposed to decrease the chance of collision and increase task performance; based on that, an optimal path is produced by using the θ-PSO path planning algorithm. Their work differs form ours as they use just a limited number of reconfigurations. They only focus on alignment, rotation and shrinkage, while our proposal is able to change the entire topology of the formation.

Other works use an approach which is called flocking. Flocking is a behaviour that is common in nature, for instance in a group of fish, birds or insects. It consists of a few basic rules that are applied to each entity of the group. When those rules are respected, the group will stay united without collisions between the group elements. There are various methods to achieve a flocking behaviour for a group of UAVs, as discussed below.

In the work by Ming Chen et al.[8] a flocking model for an UAV network based on swarm intelligence is presented. In their work they propose a set of rules to make sure that the slaves will follow the master while maintaining a certain safe distance from the master. They cannot get too close because this behaviour increases the chances of collisions; also, they cannot get too far away, because otherwise communication will be lost. Simulation results show that their model can guarantee connectivity between nodes, and it will also improve bandwidth usage.

Victor Casas et al. [9] developed a flocking model without the use of a master-slave model. The UAVs in the swarm regularly broadcast and receive movement information. That information is then used to calculate two forces: a *flock goal force*, which guides the flock towards the target location and aligns the swarm members, and a *flock members force*, which provides cohesion and separation to the flock. Those two forces are used to update a direction vector which points towards the target location, while at the same time avoids collisions. Their model is tested in simulation and in real experiments which show that a collision-free flight is ensured. They tested the model under various speeds, although all of them were rather slow (a maximum of 3m/s). Results also showed that, during real experiments, the minimum distance between UAVs is decreased; according to the authors, this is due to GPS inaccuracy.

Yazhe Tang et al. [10] presented a swarm flocking scheme that was able to work in a radio silent environment. In contrast to many other works, their approach was not based on sending (GPS) information between the swarm elements. They used two types of vision sensors (standard and thermal cameras) to track their leader, and a LiDAR sensor to sense the surrounding environment for navigation and obstacle avoidance. Because they used various high-end sensors, their flocking mechanism can be used both during the day and during the night. Indoor and outdoor experiments performed in obstacle rich environments have proven the effectiveness of the proposed method. Furthermore, their software is implemented in the robot operating system (ROS) [11], which promotes reusability through its modular design.

While flocking mechanisms are great to keep a swarm of UAVs organized, they do not provide the flexibility to completely define and change the formation itself. In many applications, it is useful to change the formation (for instance, from a line to a circle); however it is difficult to encapsulate such behaviour using flocking mechanisms. Therefore, in our work, we specifically focus on changing between different flight formations. Hence, instead of using a flocking mechanism, we propose a master-slave model where the master instructs the slaves how to safely accomplish the reconfiguration.

III. Proposed mechanism

The aim of this work is to reconfigure a swarm of UAVs, seamlessly switching from one flight formation to another. In our approach we make use of a master-slave pattern. The master is elected before taking off, as described in our previous work [12]. The master is in charge of the main calculations, and keeps the swarm synchronized throughout the different stages of the reconfiguration. All the stages are described in Figure 1. The protocol starts with the UAVs taking off and following a mission. The reconfiguration will start upon a trigger event, which can be an user input or an event predefined in the ground control station. The reconfiguration itself is divided into two stages: an analysis step where the calculations are done, and a mobility step where the UAVs move to their target locations in an intelligent manner to avoid collisions. After the swarm has reconfigured itself, the mission can continue. The protocol finishes at the end of the mission by landing all the UAVs.

A. Phase 1: Analysis

In a previous work we developed an algorithm to determine who the master should be in the scope of a UAV swarm [12]. To understand our current proposal it is only relevant to know that a single master is assigned, and that it will always be located in a central position on the flight formation to minimize losses on the wireless channel. In this first step, the master decides the slaves positions (later referred to as intelligent position). The idea is that the overall flight distance is minimised by choosing the UAV that is already closest to a new flight position to fly to it. This algorithm is also explained in more detail in [12]. Basically, it consist of the following four steps:

1) Find a central location with respect to the current location of the UAVs.
2) Calculate the euclidean distances from that central location to the positions in the new flight formation.
3) Sort this list in descending order.
4) Assign each location in the flight formation to the closest UAV.

While the algorithm was originally designed to ensure a safe and fast takeoff procedure, we were able to reuse it for our current swarm reconfiguration purposes.

In order for the master to execute this algorithm it needs to know where all the UAVs currently are, and what new locations are defined in the new swarm layout. The current

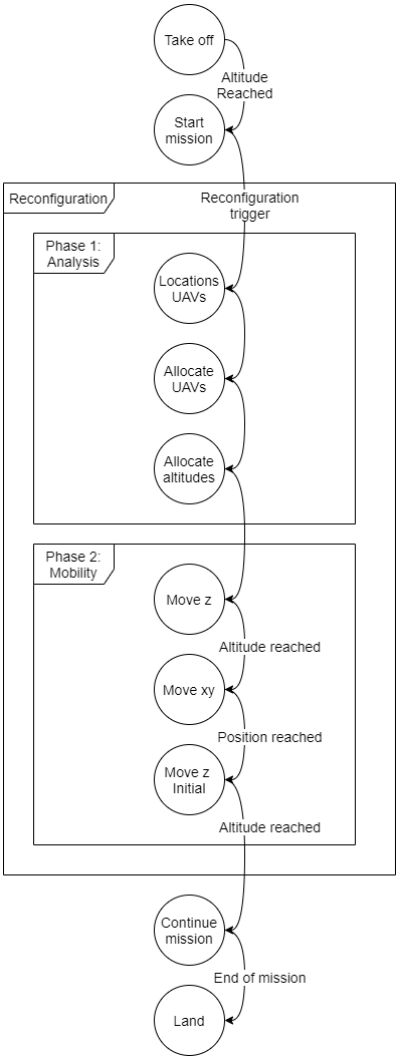


Fig. 1: Flowchart of the flight formation algorithm

locations are known by the master since it defines and maintains the swarm topology at all times. Regarding the new locations, these are associated to the new flight formation as specified by the controller. It is worth pointing out that, in the mobility stage, the UAVs will fly at different altitudes to reduce the chance of collisions. So, the next thing the master needs to do is to decide which UAV flies at which altitude. That process is fully described in Algorithm 1. It details how the master calculates (for each UAV) in what direction it has to go. Based on that direction, the UAVs are placed into different sectors. Each sector has a different altitude assigned to it. In this manner, UAVs that are likely to cross each other's path will fly at different altitudes, and thus we decrease the chances of collisions. Note that this algorithm does not guarantee a collision free reconfiguration. Once the calculations are done, the master will start sending messages with the target location (x,y,Δz) to all slaves. Upon receiving this message, the slaves will reply with acknowledgements, and, once all the slaves have received their target location, the swarm will transition to the mobility step.

Algorithm 1 Section select procedure

Require: $numberOfSections > 0$

for *UAV* **in** *UAVs* **do**
  $\Delta x \leftarrow UAV.targetLoc.x - UAV.startLoc.x$
  $\Delta y \leftarrow UAV.targetLoc.y - UAV.startLoc.y$
  $\alpha \leftarrow atan2(\Delta y, \Delta x)$
  if $\alpha < 0$ **then**
    $\alpha = \alpha + 2 \times \pi$
  end if
  $sectorWidth = \frac{2 \times \pi}{numberOfSections}$
  $sector \leftarrow 0$
  for i in range(0, *numberOfSections*) **do**
    $min \leftarrow i \times sectorWidth$
    $max \leftarrow (i+1) \times sectorWidth$
    if $min \leq \alpha < max$ **then**
      $Sector = i$
    end if
  end for
end for

B. Phase 2: Mobility

The mobility step is split up into three states: first the UAVs will change altitude, depending on his sector as explained in the previous section (movement in the Z direction), then they will go towards their target location (X,Y movement), and finally they will return to their initial altitude (return to default Z value). In each state the master will send messages to the slaves. When a slave receives the message it will perform the movement and reply with an acknowledgement once the movement is finished. The master receives the acknowledgements and, when all the slaves have sent an acknowledge message (and the master has reached its position), the master will transition to the next state. At that moment, the master will start sending messages from his new state; slaves will receive those messages, and transition too. The messages sent by the master only contain an id which represents the current state. They do not have to contain the location information because this was already sent in phase 1.

As a final remark, it is worth pointing out that our proposal is computational efficient. Algorithm 1 is the only element with significant computational requirements, and it limited to a O(N^2). Since in most practical applications the number of UAVs in a swarm will be low (below 100), this algorithm can be easily executed on the UAV's onboard computer, such as a Raspberry Pi. Also the network will not be overloaded since the message payloads are quite small.

IV. Experimental settings and results

We performed a wide set of experiments in our own UAV emulator/simulator in order to assess the validity and robustness of our proposed mechanism. Before providing a detailed explanation about our experiments and the results obtained, we will briefly discuss our simulator environment called ArduSim.

A. Ardusim

ArduSim is multi-UAV flight simulator/emulator; it is available online [13] under the Apache License 2.0. The simulator has many features, which are fully explained in our previous work [14]. Here, we will just highlight some of the key characteristics.

First of all, ArduSim makes it easy, fast and reliable to deploy a protocol that was developed in the simulator to real UAVs. It does this mainly by implementing the same open source protocols and standards that are used by the majority of the UAVs. Besides that, ArduSim really is a multi-UAV flight simulator; it is able to scale up to 100 UAVs in real time, and up to 256 UAVs in soft real time on a high-end PC (Intel Core i7-7700, 32 GB RAM). Wireless communication models, based on real experiments, are implemented to support UAV-to-UAV communications; notice that this is a basic requirement for nearly all swarm applications. Furthermore, a lot of basic UAV functionality (such as taking off, moving to a GPS location, etc.) is provided by the Application Programming Interface (API). The user is provided with a functional GUI and extensive logging features.

Overall, ArduSim is a versatile tool that provides researchers the opportunity to quickly develop new applications and protocols, without losing accuracy and/or customization.

B. Safety analysis

Our approach combines an intelligent UAV assignment (see Section III-A) with a sectorization procedure that groups UAVs moving with similar directions so that their mobility takes place at different heights (see Section III-B). To assess the effectiveness of this combined approach, we will compare it to other (simpler variants) where such mechanisms are not used, so that we can evaluate which part has the most influence and if our approach (as a whole) is effective. Therefore, we propose three other (but similar) approaches:

A. Random position assignment, no altitude change.
B. Random position assignment, different altitudes.
C. Intelligent positioning, no altitude change.
D. Intelligent positioning, different altitudes.

In our first set of experiments, 9 UAVs changed from a linear formation towards a compact mesh formation (see Figure 2 for an example). The minimum distance between UAVs in that formation was set to 10 meters, the number of sectors was equal to three and the altitude difference between sectors was of 5 meters. During the experiments we measured the time that the UAVs spent in each state (Move_Z, Move_XY, Move_Z_Initial), the minimum distance between the UAVs during the Move_XY state, and the potential number of collisions. A collision happens when the euclidean distance between two UAVs in our experiments is smaller than 5 meters to account for the GPS offset error.

The results are shown in Table I and Table II. Our experiments have shown (as stated before) that merely changing the formation layout without adopting any type of strategy is very dangerous, and prone to cause collisions. We can also observe

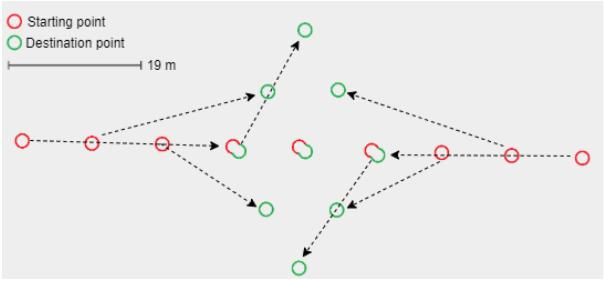


Fig. 2: Transition of 9 UAVs from a linear formation to a compact mesh

that just by changing the altitude or the position assignment of the UAVs in an intelligent manner is not enough to avoid collisions in all cases. Only when both where used could collisions be entirely avoided. Furthermore, while changing the altitude does make the process safer, an additional time overhead is introduced. The time overhead depends on the number of sectors and the altitude difference between the sectors, the impact of both parameters are discussed in more detail in the following experiments. Implementing a intelligent positioning system reduces the overall flight distance and, therefore, flight times are slightly shorter in experiments C and D.

TABLE I: Collisions and minimum distance analysis.

Ex	Nr. collisions	Min. Distance between UAVs
A	4	0.44
B	2	0.33
C	2	3.58
D	0	6.15

TABLE II: Time UAVs spend in each state.

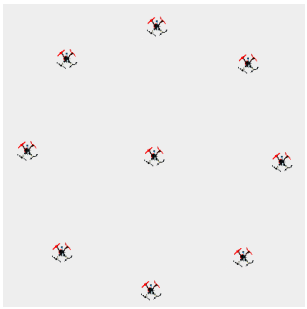
Ex	Move z [ms]	Move XY [ms]	Move Z ini [ms]
A	404	13607	400
B	6802	13030	7980
C	380	12425	400
D	8600	12415	8600

C. Scalability

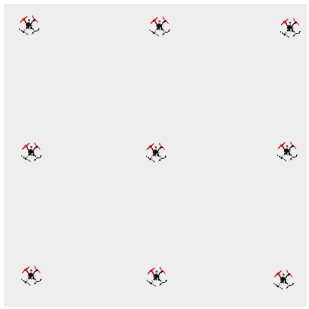
In our second experiment we want to evaluate the scalability of our protocol. We searched for the minimal number of sectors needed to complete a collision-free reconfiguration, for different number of UAVs, and for different formations (see Figure 3). All of the formations where prone to collisions due to the small distance between the UAVs that was defined ($\leq$ 10m). We started with 9 UAVs (as in the previous experiment), and increased this value up to 25 UAVs. The results are shown in Figure 4. As expected, the minimum number of sectors required to guarantee a collision-free reconfiguration increases with the number of UAVs. The rate of increase depends highly on the type of formation.

D. Differences between various transitions

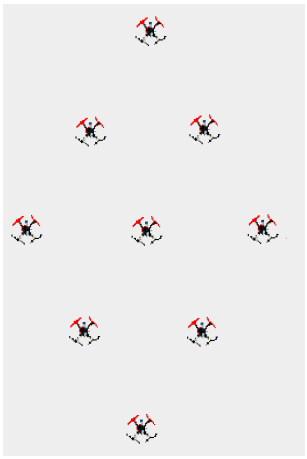
Due to our findings in the previous experiment, we investigated the influence of the type of formation in greater detail.



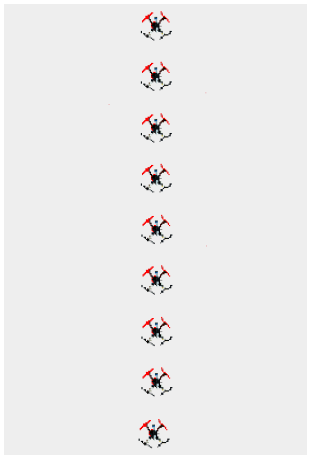
(a) Circle formation.



(b) Matrix formation



(c) Mesh formation



(d) Linear formation

Fig. 3: Different types of formations.

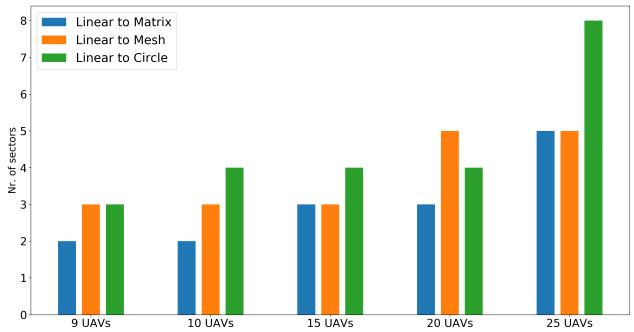


Fig. 4: Minimum number of sectors required for a collision-free reconfiguration procedure.

In particular, we tested all the possible transitions between the four flight formations considered (Linear, Matrix, Mesh, Circle). The experimental settings are similar to the previous ones. We worked with 15 UAVs in formations where the distance between the UAVs is less than 10 meters. During the experiment we searched for the minimal number of sectors needed to complete a collision-free reconfiguration. We also measured the time spent at each state. Results are shown in Figures 5 and 6. As we can observe from Figure 5, results vary significantly depending on the specific transition; in some cases, such as going from a mesh to a matrix formation, just a few sectors are needed. In other cases (e.g. matrix to linear) the angles α calculated in Algorithm 1 are very similar, and so many sectors are required in order to separate the UAVs in different altitude groups. In the presence of many groups, the target altitude can grow a lot, resulting in a high time overhead (in the worst-case scenarios), as shown in Figure 6. Due to the similar shape of both figures, we can see the correlation between the number of sectors and the overall reconfiguration time. Furthermore, we can conclude that the time spend moving in the xy-plane fluctuates only a little, being limited to a maximum of 6.8 seconds in our experiments.

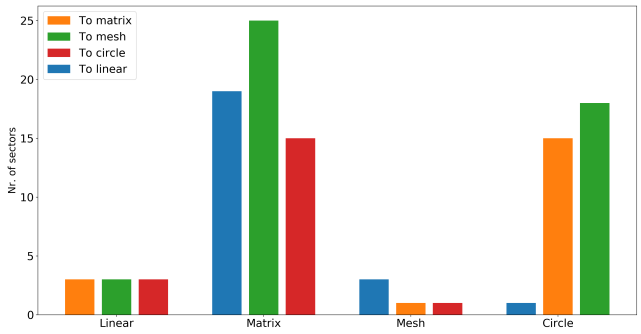


Fig. 5: Minimum number of sectors required for a collision-free reconfiguration) w.r.t. the type of transition.

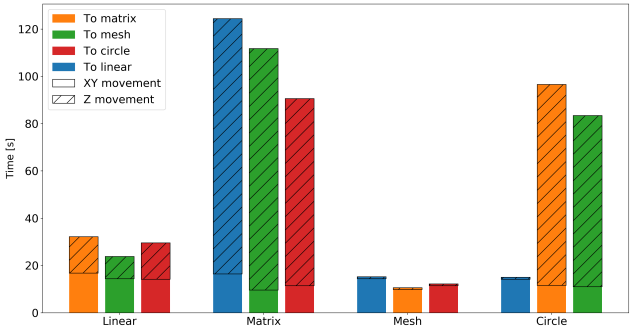


Fig. 6: Time spent moving horizontally (state 2) and vertically (states 1,3).

E. Time overhead

Finally, we further investigated the time overhead introduced by changing UAV altitudes during reconfiguration. To achieve this we start by finding the value of the one-way delay T for which:

$$\int_0^T v(t)dt = D \tag{1}$$

where

$$D = num.sectors \times sectors\_offset$$

This one-way delay refers to both upward or downward movements. We can approximate this one-way time overhead T as:

$$T \cong \frac{D}{\hat{v}_{0\to T}} + \epsilon \tag{2}$$

where $\hat{v}_{0\to T}$ refers to the expected speed during the entire mobility from time 0 to T, and ϵ accounts for the additional time associated to acceleration and deceleration processes.

In our experiments $\hat{v}_{0\to T}$ was set to 2 m/s, and the distance between the sectors ($sector\_offset$) to 5 meters. The number of sectors ranged between 2 and 8. Figure 7 compares the estimated time overhead for the two-way vertical mobility against the real time overhead measured in our experiments. We can clearly observe a linear pattern (as suspected by the derivation), and in our case the average value of ϵ is of 1.5s.

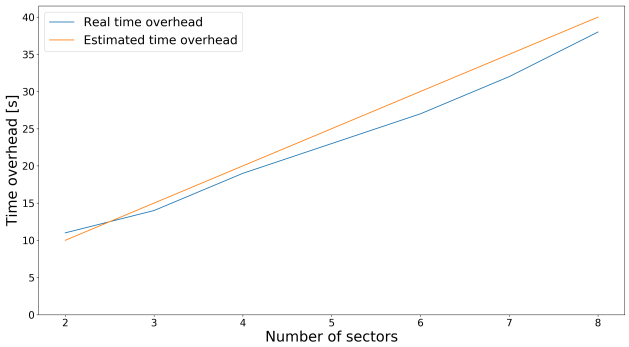


Fig. 7: Estimated time overhead vs real time overhead

V. Conclusions and Future Work

Research in the field of Unmanned Aerial Vehicles (UAVs) has been significantly boosted over the last few years, and as a result we are now able to tackle more difficult problems such as multi-UAV coordinated flights. These so called swarms are able to extend UAV-based applications by allowing work to be done in parallel, with more redundancy, and providing the ability to carry heavier loads. However, coordinating multiple UAVs is not an easy task. In this work we focused specifically on the reconfiguration of a swarm. This is a relevant behaviour that can be used to make many applications more efficient and/or effective. However, the chances of collision become high during reconfiguration, and so it becomes an issue that must to be dealt with. Our proposal is based on an intelligent position assignment system that reduces the chances of flight paths crossing during formation reconfiguration. The chances of collision are further reduced by distributing the UAVs over different altitude levels during the reconfiguration period. This simple, computationally efficient approach can be easily applied to various environments. However, it is not able to fully guarantee a collision-free reconfiguration in all cases, and when scaled-up to many UAVs the time overhead introduced becomes significant. For these reasons, our future work will focus on more complex algorithms that combine path prediction with machine learning approaches to avoid collisions in a more timely efficient manner.

Acknowledgments

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] G. S. Research, "Drones: Reporting for work." https://www.goldmansachs.com/insights/technology-driving-innovation/drones/, 2014. Accessed: 2020-06-04.

[2] 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. 48572–48634, 2019.

[3] S. Hayat, E. Yanmaz, and R. Muzaffar, "Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint," *IEEE Communications Surveys Tutorials*, vol. 18, no. 4, pp. 2624–2661, 2016.

[4] P. Vincent and I. Rubin, "A framework and analysis for cooperative search using uav swarms," in *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, (New York, NY, USA), pp. 79–86, ACM, 2004.

[5] M. Aljehani and M. Inoue, "Multi-UAV tracking and scanning systems in M2M communication for disaster response," in *2016 IEEE 5th Global Conference on Consumer Electronics*, pp. 1–2, Oct 2016.

[6] 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, p. 100106, 2019.

[7] V. T. Hoang, M. D. Phung, T. H. Dinh, Q. Zhu, and Q. P. Ha, "Reconfigurable multi-uav formation using angle-encoded pso," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pp. 1670–1675, 2019.

[8] M. Chen, F. Dai, H. Wang, and L. Lei, "Dfm: A distributed flocking model for uav swarm networks," *IEEE Access*, vol. 6, pp. 69141–69150, 2018.

[9] V. Casas and A. Mitschele-Thiel, "Implementable self-organized flocking algorithm for uavs based on the emergence of virtual roads," in *Proceedings of the 6th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, DroNet '20, (New York, NY, USA), Association for Computing Machinery, 2020.

[10] Y. Tang, Y. Hu, J. Cui, F. Liao, M. Lao, F. Lin, and R. Teo, "Vision-aided multi-uav autonomous flocking in gps-denied environment," *IEEE Transactions on Industrial Electronics*, vol. PP, pp. 1–1, 04 2018.

[11] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system."

[12] 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.

[13] "ArduSim. accurate and real-time multi-UAV simulation." https://bitbucket.org/frafabco/ardusim/src/master/, 2017. Accessed: 2020-05-11.

[14] 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, pp. 170–190, sep 2018.