Formation Maintenance and Collision Avoidance in a Swarm of Drones

Jawad Naveed Yasin
University of Turku
Vesilinnantie 5
20500 Turku,
Finland
janaya@utu.fi

Mohammad-Hashem Haghbayan
University of Turku
Vesilinnantie 5
20500 Turku, Finland
mohhag@utu.fi

Jukka Heikkonen
University of Turku
Vesilinnantie 5
20500 Turku,
Finland
jukhei@utu.fi

Hannu Tenhunen
University of Turku
Vesilinnantie 5
20500 Turku, Finland
hannu.tenhunen@utu.fi

Juha Plosila
University of Turku
Vesilinnantie 5
20500 Turku, Finland
juplos@utu.fi

ABSTRACT
Distributed formation control and obstacle avoidance are two important challenges in autonomous navigation of a swarm of drones and can negatively affect each other due to possible competition that arises between them. In such a platform, a multi-priority control strategy is required to be implemented in every node in order to dynamically optimise the tradeoffs between collision avoidance and formation control w.r.t. given system constraints, e.g. on energy and response time, by reordering priorities in run-time and selecting the suitable formation and collision avoidance approach based on the state of the swarm, i.e., the kinematic parameters and geographical distribution of the nodes as well as the location of the observed obstacles. In this paper, we propose a method for formation/collision co-awareness with the goal of energy consumption and response time minimisation. The algorithm consists of two partial nested feedback-based control loops and based on three observations: 1) for formation maintenance the relative location of the neighbour nodes; 2) observation of an obstacle by a local sensor, represented by a boolean value, used for both formation control and collision avoidance; and 3) in critical situations for avoiding collisions, the distance of an obstacle to the node. The obtained comprehensive experimental results show that the proposed approach appropriately keeps the formation during the swarm's travel in the presence of different obstacles.

CCS concepts
•Formation Maintenance~Formation Maintenance →Collision aware formation~Swarm Robotics→Collision avoidance~Swarm Robotics→Autonomous system~Robotics

Keywords
Swarm Formation; Multi-robot Systems; Swarm Intelligence; Collision Avoidance; Autonomous Vehicles; Autonomous Aerial Vehicles

ISCSIC 2019, September 25–27, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-7661-7/19/09…$15.00
https://doi.org/10.1145/3386164.3386176

1. INTRODUCTION
Optimisation in different aspects, such as in resource allocation/utilisation, of autonomous navigation for a swarm of drones (i.e. UAVs) is gaining traction in the research community. That is due to lack of multi-objective strategies in the operation of traditional UAVs, such as achieving different system goals optimally or even near-optimally under several design/mission limitations such as flight time and payload [16, 17].

Two important problems in navigation of a swarm of drones are the formation of the swarm (and its maintenance) and collision avoidance [25, 5]. The focus of collision avoidance is mainly on path planning of individual drones in order to avoid possible collisions between drones themselves and between drones and external obstacles in the environment [5], while the location of each drone is defined w.r.t. the other drones in a formation [15]. Algorithms for formation control can be categorised into three general approaches [4, 23]: 1) the leader-follower based approach, in this approach every drone function autonomously by adjusting and maintaining its coordinates, with respect to the leader, in the formation as accurately as possible to other drones [6, 10, 12, 13]; 2) the behaviour based approach, in this approach based on a pre-defined strategy, a drone selects one of the numerous/multiple behaviours [21, 11]; 3) the virtual structure based approach, in this approach drones of the formation are navigated through the same trajectory as if there was a single big drone [14, 19, 20, 22]. Among the mentioned approaches, the leader-follower based method is more common due to its ease of analysis and implementation [1, 7, 24].

Collision avoidance algorithms can be categorised into three general classes: 1) optimisation based methods, these methods rely on static obstacles with known positions and dimensions for calculating and efficient route within a finite time horizon. They are mainly focused on determining the optimal or near-optimal solutions for path planning and motion characteristics of each drone w.r.t. the other drones and obstacles [8, 18, 27]; 2)

force-field based methods, these are inspired by attractive/repulsive electric forces that exist among charged objects [3, 9]; and 3) sense-and-avoid based methods have short response times as they are primarily focused on reducing required the computational power. It is achieved by simplifying the collision avoidance process to individual detection and avoidance of obstacles [2, 26].

To maintain a formation, we need to consider collision avoidance, and to avoid collisions, we need to consider the intended formation. The algorithm designed in this paper takes into account the dynamic swarm formation process together with an efficient collision avoidance scheme, as well as a tight queue formation setting with variable speeds of UAVs. Our approach focuses on integrating all these features together. If the process is done autonomously, then we need to consider how formation control and collision avoidance can be merged together efficiently. In dynamic formation control and collision avoidance, there are many unforeseen factors/parameters that affect optimal implementation of the algorithm. For example, unpredicted obstacles or narrow openings between obstacles may force a formation change by prioritising collision avoidance over formation maintenance.

The rest of the paper is organised as follows. In Section 2, the development of the proposed algorithm is described. Section 3 focuses on simulation and results. Lastly, Section 4 presents some concluding remarks.

2. THE PROPOSED APPROACH

The general pseudo code of the top-level formation/collision avoidance co-awareness algorithm is given in Algorithm 1. We assume that the UAVs are spawned at random coordinates, and assignment of IDs to the nodes is initiated before the mission is started. Based on the assigned IDs, each node is executing this top-level algorithm locally, utilising its available on-board processing units. Navigation of a swarm is based on path planning for the leader(s) and is out of the scope of this work. Algorithm 1 initiates formation maintenance by getting feedback on the distance the node has to its neighbours (Line 2 in Algorithm 1). After this, a formation error, i.e., $E_{formation}$, is calculated. To do that, a threshold value is added to the absolute value of the relative angular and distance position of a drone w.r.t. its neighbours (Line 4). The threshold value determines the degree of freedom each drone can have in the formed swarm. The formation error is processed differently depending on whether an obstacle is detected or not. If any obstacle is observed, a collision error is calculated, indicating the difference between the obstacle distance and collision radius (Line 5). The algorithm works on three cases (Lines 8-13). If any obstacle is detected and only a positive formation error exists, then the algorithm reconfigures the relative angular and distance position of the drones via the *Formation* function to decrease the error (Lines 8-9). In the case where an obstacle is detected by the sensors and $0 < E_{formation}$, then the *Collision-aware formation* function is executed, reconfiguring the formation of the swarm by considering the location of the observed obstacle (Lines 10-11) However, if $0 < E_{collision}$, the obstacle is too close to the drone, and the *Collision avoidance* function is launched (Lines 12-13). This considers only the distance of the drone w.r.t. the obstacle and is oblivious of the formation. Notice that if $E_{formation} \leq 0$, then the algorithm does not change the formation, i.e., only *Collision avoidance* of the above three functions can be executed. As a fail- safe check, if the drone's leader gets out of range or is not detected, the drone temporarily sets itself as its own leader and starts navigating towards the destination (Lines 14-15).

However, at the moment the leader comes back in the visible range, the drone immediately starts to follow it and comes back to the formation (Lines 16-17). Each of the functions in Algorithm 1 is explained in the following subsections in more detail.

Algorithm 1 Formation/collision co-awareness

1: **procedure** Formation Collision Co-awareness
2: formation ← Initiate Formation;
3: **while** True **do**
4: $E_{formation}$ ← Calculate distance and angular formation error (formation);
5: $D_{obstacle}$, $E_{collision}$ ← Calculate obstacle distance and collision error if any;
6: STATE ← ($0 < E_{formation}$, Obstacle is probeable by local sensors, $0 < E_{collision}$);
7: **switch** STATE **do**
8: **case** (True, False, False)
9: Formation ($E_{formation}$);
10: **case** (-, True, False)
11: Collision-aware formation ($E_{formation}$,$D_{obstacle}$);
12: **case** (-, -, True)
13: Collision avoidance ($E_{collision}$);
14: **if** MyLeader is out of range **then**
15: MyLeader ← Self;
16: **else**
17: MyLeader ← Leader(Self);

2.1 Formation Algorithm

Algorithm 2 Formation

1: **procedure** Formation (($R_{distance}$, $R_{angular}$) ←$R_{formation}$, ($E_{distance}$, $E_{angular}$) ←$E_{formation}$)
2: (*distanceError*, *angularError*) ←$R_{formation}$-$E_{formation}$
3: **if** $0 < ABS(distanceError)$ **then**
4: Calculate Distance From Respective Leader;
5: Accelerate/decelerate Accordingly;
6: **if** $0 < ABS(angularError)$ **then**
7: Set Angular Direction ($R_{angular}$);

Each node in a swarm has a unique ID to be recognizable from the other nodes and the leader. The leader navigates towards the assigned destination, and the followers try to maintain the formation while following the leader by keeping the required distances for the formation. Algorithm 2 shows the general pseudo code of the formation function. The input of the algorithm comprises the pre-specified formation information that is considered the reference, i.e., $R_{formation}$, and the instantaneous captured formation information, i.e.,$E_{formation}$, that is the estimated distance and angle each drone has w.r.t. its neighbours (e.g. the leader and follower in a queue formation) at a given time. First, the angular and distance errors are calculated, showing the deviation of each drone w.r.t. its pre-specified position (Line 2 in Algorithm 2). If the distance or angular error is larger than a threshold, indicating that the amount of deviation is too high and negatively affects the formation, the algorithm starts to change the state of the swarm by manipulating the node's mechanical actuators. The *distanceError* is calculated as follows:

$$distanceError = \sqrt{(L_x - F_x)^2 + (L_y - F_y)^2} \qquad (1)$$

where L_x, L_y are leader's x and y-coordinates and F_x, F_y are the follower's. In the case of a positive distance error, meaning that the distance of the node to its neighbour is too high, the drone accelerates to catch up and cover the distance. Similarly, when

the distance error is negative, which means the node is too close to the neighbour, then the drone starts decelerating to decrease the absolute value of the distance error. A local speed control is used in each drone for the possible manoeuvres for maintaining the desired distance and orientation.

The angle of each leader drone is determined by the follower as shown in Figure 1 and is calculated as follows:

$$Angle(y,x)=\begin{cases}\arctan(\frac{y}{x}) & if_x>0,\\ \arctan(\frac{y}{x})+\pi & if_x<0\ and\ y\geq 0,\\ \arctan(\frac{y}{x})-\pi & if_x<0\ and\ y<0,\\ +\frac{\pi}{2} & if_x=0 and\ y>0,\\ -\frac{\pi}{2} & if_x=0 and\ y<0\end{cases} \quad (2)$$

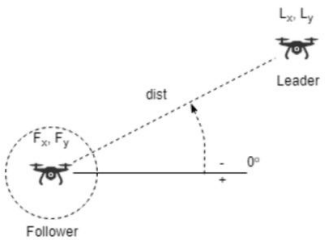


Figure 1. Distance and Direction Calculation.

The result is always between $-\pi$ and π. The angle is formed w.r.t. the positive X-axis by using the plane vector from the origin to the target. Since the signs of both inputs are known and taken into consideration, the correct quadrant of the computed angle, i.e., *angularError*, and the corresponding direction values are calculated by using the result from the above equation as follows:

$$angularError = (dx = \cos(Angle), by = \sin(Angle)) \quad (3)$$

2.2 Collision-aware Formation Algorithm

In the cases where keeping the formation is not possible due to detection of an obstacle by one of the nodes, the proposed collision-aware formation function is selected in Algorithm 1. This function is specified in Algorithm 3. The main strategy in such situations, where the obstacle's distance $D_{obstacle}$ is within the range of a node but not small enough to justify actual collision avoidance measures, is to symmetrically shift the swarm and reshape the formation at the same time in order to divert the observer node, and thereby its followers, from the potential collision course. Based on this strategy, the algorithm first defines a new formation, close to the original one, which makes bypassing the obstacle easier (Line 2 in Algorithm 3). After this, the new formation is fed to the formation function to reshape the swarm (Line 3). In the next iteration of Algorithm 1, the collision error and obstacle distance will be recalculated, and if the obstacle is not anymore within the observation range of the node, the formation returns back to the original one (Lines 8-9 in Algorithm 1).

Algorithm 3 Collision-aware Formation Algorithm

1: **procedure** Collision-aware Formation($E_{formation}$, $D_{obstacle}$)
2: *newTemporaryFormation* ← determineNewFormation ($E_{formation}$, $D_{obstacle}$);
3: Formation (*newTemporaryFormation*);

2.3 Collision Avoidance Algorithm

Upon encountering an obstacle critically close to the node, the collision avoidance function is invoked. Its pseudo code is shown in Algorithm 4. It starts detecting the edges ofthe obstacle, at which point the angles between the node's and obstacle's positions as well as the exact position of the obstacle can be determined (Line 2 in Algorithm 4). If there are visible edges, it is checked whether there are multiple obstacles or only one obstacle in front of the node (Lines 3-4). If there are multiple obstacles in vicinity, the gap between the obstacles is calculated (Line 5), see Figure 3(a). Based on the gap calculations, the algorithm decides whether it is safe for the drone to go through the detected gap (Lines 6-7). If not, the drone needs to go around the obstacles to avoid collisions by treating the obstacles as a single entity (Lines 8-10). Moreover, if only one obstacle is detected (Line 11), the short-term path planning is done accordingly by calculating in which direction the node should navigate to bypass the obstacle with minimal deviation from its original path (Lines 12-13). However, if the obstacle's neither corner is visible to the on-board sensor system of the UAV (Line 14), a tangent line from the drone's current position to the destination coordinates is determined, and the path is followed accordingly by going towards the destination, staying as close to the tangent line as possible while avoiding the obstacle (Lines 15-16), see Figure 3(b).

Algorithm 4 Collision Avoidance Algorithm.

1: **procedure** Collision Avoidance($E_{collision}$)
2: *edges* ← detect edges ();
3: **if** *edges* contains visible edges **then**
4: **if** More than one obstacle is detected **then**
5: *Dfs* ← Calculate gap between obstacles (*edges*);
6: **if** *Dfs* > *Rc* **then**
7: Short-term path planning (*edges*); . Align UAV to pass through the obstacles
8: **else**
9: *pathPlan* ← Calculate path plan (*edges*);
10: Short-term path planning (*pathPlan*);
11: **else**
12: *pathPlan* ← Calculate path plan (*edges*);
13: Short-term path planning (*pathPlan*);
14: **else**
15: *tangentLine* ← Calculate tangent line();
16: Short-term path planning (*tangentLine*);

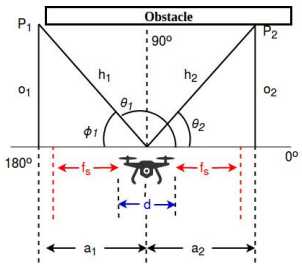


Figure 2. Obstacle Detection and Avoidance using geometric guidance law.

To summarize, the proposed collision avoidance function

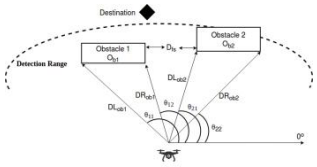
(Algorithm 4) operates based on three diffffferent cases, namely: 1) there is one obstacle and its edges are detected, Figure 2; 2) there are two or more obstacles and their edges and gaps are detected, Figure 3(a); and 3) there are no visible edges (i.e. the obstacle is very large), in which case the path is determined based on a tangent line, Figure 3(b). When the distance from the UAV to the obstacle, $D_{obstacle}$, is within the middle ground of the detection range and safe distance f_s (Table 1, Figure 2), the drone starts to decelerate whilecalculating the angle at which the detected obstacle lies in order to deviate from that path. As soon as $D_{obstacle}$ gets closer to f_s, the collision avoidance algorithm takes complete control and guides the drone to bypass the obstacle safely.

Figure 2 shows the case where the node detects the edges of a single obstacle. The variables and their explanations are given in Table 1.

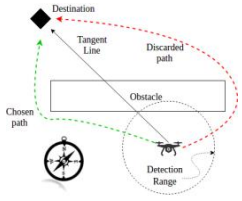
Table 1. Description of Variables from Figure 2.

Variables	Description
d_u	the width of the UAV
f_s	minimum safe distance allowed between the obstacle and the outer edge of the UAV
h_1 h_2	diagonal detected distance of the object(edges) on the left and right sides respectively
a_1 a_2	distance of the object on the left and right side of the UAV respectively, calculated from the centre of the UAV
o_1 o_2	distance of the object straight ahead of the UAV
P_1 P_2	point where the object/edge was detected by the US sensor
θ_1 θ_2	angles of the edges
φ_1	linearised angle

Once the exact coordinates of the object have been calculated, all possible combinations of the object's location can be analysed and the decision is made accordingly. The algorithm decides whether to continue along the current path or whether the UAV must be diverted to a certain extent to avoid a collision with the obstacle. For example, considering Figure 2, if $a_2 < (du/2)+f_s$, the UAV will be diverted to the left from its current path as continuing along the same path may result in a collision with the obstacle on the right side of the drone. On the other hand, if $a_2 \geq (du/2)+f_s$, no deviation from the current path is needed due to the obstacle on the right side.



(a) Multiple Obstacles with opening between them.



(b) Local decision making in case of unknown length of obstacle.

Figure 3. Multiple Objects and Local decision making Figures.

When detecting multiple obstacles, and their corresponding edges, two different actions can be performed, i.e.: 1) extending the collision envelope when $Df_s \leq R_c$, and 2) activating detection and gap calculation when $Df_s > R_c$. Here Df_s is the distance between the inner edges of two obstacles and R_c is the collision radius. Based on the action 1, the collision envelope is extended including both obstacles that are lying in close proximity and considering them a single object while calculating the avoidance maneuver parameters (Lines 9-10 in Algorithm 4). In this case, θ_{11} and θ_{22} are taken into account, as shown in Figure 3(a), and the two obstacles are treated as a single entity. Calculations are done to bypass the combined obstacle by flying either from its left or right side.

In the case of the action 2, the algorithm analyses the opening between the obstacles to find out if its width Df_s satisfies the condition $Df_s \geq 2f_s + du$, where f_s is the safe distance and d_u is the width of the UAV (Table 1, Figure 2), indicating that the drone has enough space to pass through the gap between the obstacles. If this condition is not met, it is physically not possible for the drone to safely go through the gap, i.e., the gap is too narrow and does not provide a sufficiently large safety margin. Therefore, in this situation, the operation switches to the action 1. In other words, instead of considering the angles θ_{12} and θ_{21} like in the action 2, the angles θ_{11} and θ_{22} are taken into account (Figure 3(a)), and the obstacles are considered as a single combined object which is bypassed either from its left or right side.

In the case where the dimensions of the obstacle are not known, as the obstacle extends beyond the detection range of the sensor system, a tangent line is drawn to the destination, using the data provided by the local GPS unit, that is the line showing the shortest path between the node and the destination (Figure 3(b)). Based on the angle of the tangent line, the node decides which direction it should take. For example in Figure 3(b), among the two possible choices that are west/left and east/right, using the angle of the tangent line, it is determined that the destination lies to the north-west of the UAV, and hence the green path is chosen while avoiding the obstacle by staying at the minimum safe distance from the obstacle for safe maneuvering.

3. SIMULATION & RESULTS

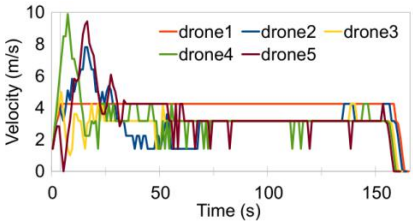
A 700x500m 2D-XY area is used as the basis for simulations, i.e., all the objects are considered to be at a constant altitude. The number of drones is set to 5, and the IDs assigned to them are from 1 to 5. UAV1 is chosen as the leader of the swarm. The formation is set to be a simple queue formation, which means that the rest of the UAVs in the swarm are following their immediate leaders, i.e. UAV2 follows UAV1, UAV3 follows UAV2, and so on.

As soon as the drones are spawned, the main leader UAV1 starts moving towards the destination, and UAVi + 1 starts to track UAVi, i = 1..4, where each follower/tracker calculates its distance to its respective leader, accelerating/ decelerating (when needed) to reach and maintain the desired distance to its leader.

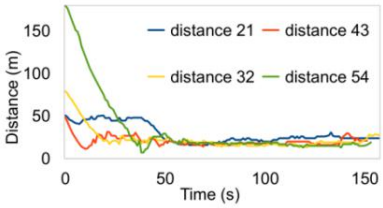
To analyse performance and efficiency of the proposed algorithm, we report and compare some obtained measurements of the system that is based on our experiment:

Figure 4(a) and 4(b) show the relative distances and velocities of all the UAVs. The relative distance is with respect to the local/immediate leader. As can be seen, after the warm-up phase, the relative distances and velocities mainly remain constant, which shows that the proposed algorithm reliably maintains

tracking and safe distance between each leader and follower in the swarm, avoiding significant fluctuations in distances and velocities.



(a) Distance of each drone from its respective leader



(b) Velocities of all UAVs

Figure 4. Distance and Velocity graph of the UAVs.

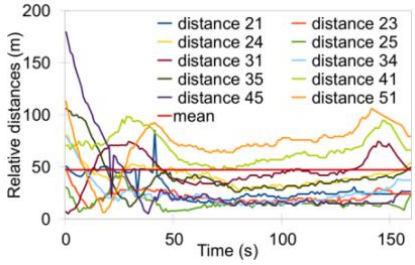


Figure 5. Relative distance of all drones from each other.

Figure 5 shows the relative distances between each UAV and all the other UAVs. This is a metric to show how well the formation is being kept. As the graph shows, initially the UAVs are at different (semi-random) coordinates and then start the formation process by getting close to their respective leaders, maintaining the minimum required distance. The standard deviations calculated before and after the drones have reached the queue formation are given in Table 2. The values show that the proposed algorithm maintains the formation tightly, without significant variation in distances, after the UAVs have reached the desired formation. In the table, σ_1 is the total standard deviation of the drones' distances from the starting point when the nodes are spread out, coming into the formation, until they reach the destination. σ_2 is the calculated standard deviation of the drones' distances from the point the nodes reach the formation until they arrive in the destination.

Table 2. Calculated σ of UAVs before and after coming into Formation.

UAV No.	Standard Deviation before formation after formation (σ_1)	Standard Deviation before formation after formation (σ_1) (σ_2)
UAV 2	10.5430785734	7.0531634715
UAV 3	5.0412687286	3.1362290768
UAV 4	11.1664821141	3.128475562
UAV 5	36.9613886179	3.5387336333

In order to compare our proposed method, we implemented the algorithm presented in [9] and set it side by side with our method. The simulation results for distance maintenance between the nodes are shown in Figure 6, where *distance 21* represents the distance maintained between second and first nodes (similar trend is followed by other UAVs). Average distance maintained by the UAVs based on the two considered methods is shown in Figure 7. From the results presented, it is evident that as compared with the reference method presented in [9], our proposed method maintains the distance between the drones very well.

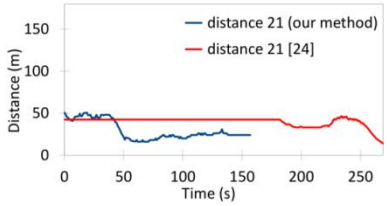


Figure 6. Comparison of distance maintenance from UAV2 to UAV1.

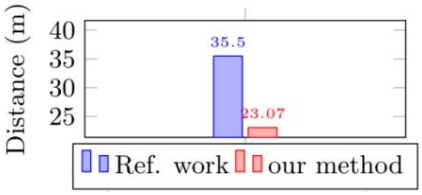


Figure 7. Average distance between UAVs.

The reference algorithm takes action when both edges of an obstacle are visible, whereas our algorithm starts taking collective measures as soon as an obstacle comes within the detection range. Furthermore, the authors in [9] have not considered alternative measures for two or more closely placed obstacles. If two obstacles are in close proximity to each other, the method always considers them a single object, even in the case where a clear gap exists between the obstacles. This leads to sub-optimal flight paths. In our algorithm, on the other hand, the detected gaps are taken into consideration. The algorithm determines if there is a sufficient space between the obstacles for the UAV to go through and takes action accordingly.

4. CONCLUSION

In this paper, we developed a formation maintenance algorithm for multiple UAVs, integrated with a collision avoidance capability. We theoretically investigated the behaviour of the proposed algorithm and tested it in a simulation environment. The simulations showed satisfactory results, demonstrating that the simulated UAVs were able to dynamically and reliably bypass obstacles without colliding with them, while maintaining the given swarm formation. By the ability to accelerate and decelerate on demand, the drones can efficiently reach their respective leaders or find their places in the formation when they start at random coordinates or wander off due to the presence of obstacles in their vicinity. Furthermore, the decentralized distribution of the algorithm allows the UAVs to take fast local decisions when in close proximity to obstacles, making the method highly robust and efficient. The collision avoidance scheme is able to flexibly handle situations with multiple detected obstacles. Moreover, the algorithm also takes care of the lost UAVs, routing them towards the destination by making a temporary formation when necessary. The multi-priority strategy works appropriately, changing the priorities of the different

parts/functions of the algorithm whenever needed. As a comparison, we showed that the efficiency of the proposed algorithm in terms of the response time, flexibility, and robustness outperforms one of the best known existing methods.

ACKNOWLEDGEMENT

This work has been supported in part by the Academy of Finland-funded research project 314048 and Finnish Cultural Foundation.

References

[1] Arshad Mahmood and Yoonsoo Kim. 2015. Leader following formation control of quadcopters with heading synchronization. *Aerospace Science and Technology*, 47, 68 –74.

[2] B. M. Albaker and N. A. Rahim. 2009. A survey of collision avoidance approaches for unmanned aerial vehicles. In *2009 International Conference for Technical Postgraduates (TECHPOS)*,1 –7. doi: 10 . 1109 / TECHPOS.2009.5412074.

[3] B. M. Albaker and N. A. Rahim. 2010. Unmanned aircraft collision detection and resolution: concept and survey. In *2010 5th IEEE Conference on Industrial Electronics and Applications*, 248–253. doi: 10.1109/ ICIEA.2010.5516808.

[4] C. B. Low and Q. S. Ng. 2011. A flexible virtual structure formation keeping control for fixed-wing uavs. In *2011 9th IEEE International Conference on Control and Automation (ICCA)*, 621–626.

[5] C. Zhuge, Y. Cai, and Z. Tang. 2017. A novel dynamic obstacle avoidance algorithm based on collision time histogram. *Chinese Journal of Electronics*, 26, 3, 522– 529.

[6] DongBin Shen, ZhenDong Sun, and WeiJie Sun. 2014. Leader-follower formation control without leader's velocity information. *Science China Information Sciences*, 57, 9, 1–12.

[7] H. Su, X. Wang, and Z. Lin. 2009. Flocking of multiagents with a virtual leader. *IEEE Transactions on Automatic Control*, 54, 2, 293–307.

[8] Hung Pham, Scott A. Smolka, Scott D. Stoller, Dung Phan, and Junxing Yang. 2015. A survey on unmanned aerial vehicle collision avoidance systems. *CoRR*, abs/1508.07723. arXiv: 1508.07723. http://arxiv.org/abs/1508.07723.

[9] J. Seo, Y. Kim, S. Kim, and A. Tsourdos. 2017. Collision avoidance strategies for unmanned aerial vehicles in formation flight. *IEEE Transactions on Aerospace and Electronic Systems*, 53, 6, 2718–2734.

[10] Jiu-Gang Dong. 2012. Flocking under hierarchical leadership with a free-will leader. *International Journal of Robust andNonlinear Control*, 23, (June 2012).

[11] Jonathan RT Lawton, Randal W Beard, and Brett J Young. 2003. A decentralized approach to formation maneuvers. *IEEE transactions on robotics and automation*, 19, 6, 933–941.

[12] Kwang-Kyo Oh, Myoung-Chul Park, and Hyo-Sung Ahn. 2015. A survey of multi-agent formation control. *Automatica*, 53, 424 –440.

[13] L Buzogany, M Pachter, and J D'azzo. 1993. Automated control of aircraft in formation flight. In *Guidance, Navigation and Control Conference*, 3852.

[14] Longfei Dong, Yangzhou Chen, and Xiaojun Qu. 2016. Formation control strategy for nonholonomic intelligent vehicles based on virtual structure and consensus approach. *Procedia Engineering*, 137, 415 –424. Green Intelligent Transportation System and Safety.

[15] Lvlong He, Peng Bai, Xiaolong Liang, Jiaqiang Zhang, and Weijia Wang. 2018. Feedback formation control of uav swarm with multiple implicit leaders. *Aerospace Science and Technology*, 72, 327 –334. issn: 1270-9638. doi: https : / / doi . org / 10 . 1016 / j . ast . 2017 . 11 . 020. http://www.sciencedirect.com/science/article/pii/S1270963816309816.

[16] M. Bowkett, K. Thanapalan, and E. Constant. 2017. Operational safety analysis and controller design of a dual drones system. In *2017 International Symposium on Computer Science and Intelligent Controls (ISCSIC)*, 82–87. doi: 10.1109/ISCSIC.2017.32.

[17] M. Campion, P. Ranganathan, and S. Faruque. 2018. A review and future directions of uav swarm communication architectures. In *2018 IEEE International Conference on Electro/Information Technology (EIT)*, 0903–0908.

[18] Nathan E Smith, Richard Cobb, Scott J Pierce, and Vincent Raska. 2014. Optimal collision avoidance trajectories via direct orthogonal collocation for unmanned/remotely piloted aircraft sense and avoid operations. In *AIAA guidance, navigation, and control conference*, 0966.

[19] Norman HM Li and Hugh HT Liu. 2008. Formation uav flight control using virtual structure and motion synchronization. In *2008 American Control Conference*. IEEE, 1782–1787.

[20] Randal W Beard, Jonathan Lawton, and Fred Y Hadaegh. 2001. A coordination architecture for spacecraft formation control. *IEEE Transactions on control systems technology*, 9, 6, 777–790.

[21] T. Balch and R. C. Arkin. 1998. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14, 6, 926–939.

[22] Tim D Barfoot and Christopher M Clark. 2004. Motion planning for formations of mobile robots. *Robotics and Autonomous Systems*, 46, 2, 65–78.

[23] Wei Ren. 2006. Consensus based formation control strategies for multi-vehicle systems. In *2006 American Control Conference*, 6 pp.–.

[24] Wenwu Yu, Guanrong Chen, and Ming Cao. 2010. Distributed leader–follower flocking control for multiagent dynamical systems with time-varying velocities. *Systems Control Letters*, 59, 9, 543 –552.

[25] X. Wang, V. Yadav, and S. N. Balakrishnan. 2007. Cooperative uav formation flying with obstacle/collision avoidance. *IEEE Transactions on Control Systems Technology*, 15, 4, 672–679.

[26] Xavier Prats, Luis Delgado, Jorge Ramirez, Pablo Royo, and Enric Pastor. 2012. Requirements, issues, and challenges for sense and avoid in unmanned aircraft systems. *Journal of aircraft*, 49, 3, 677–687.

[27] Xiaojing Zhang, Alexander Liniger, and Francesco Bor 2017. Optimization-based collision avoidance. *arXiv preprint arXiv:1711.03449*.