Navigation of Autonomous Swarm of Drones Using Translational Coordinates

Jawad N. Yasin[1](✉), Sherif A.S. Mohamed[1], Mohammad-Hashem Haghbayan[1], Jukka Heikkonen[1], Hannu Tenhunen[1,2], and Juha Plosila[1]

[1] Autonomous Systems Laboratory, Department of Future Technologies, University of Turku, Vesilinnantie 5, 20500 Turku, Finland
{janaya,samoha,mohhag,jukhei,juplos}@utu.fi

[2] Department of Industrial and Medical Electronics, KTH Royal Institute of Technology, Brinellvägen 8, 114 28 Stockholm, Sweden
hannu@kth.se

Abstract. This work focuses on an autonomous swarm of drones, a multi-agent system, where the leader agent has the capability of intelligent decision making while the other agents in the swarm follow the leader blindly. The proposed algorithm helps with cost cutting especially in the multi-drone systems, i.e., swarms, by reducing the power consumption and processing requirements of each individual agent. It is shown that by applying a pre-specified formation design with feedback cross-referencing between the agents, the swarm as a whole can not only maintain the desired formation and navigate but also avoid collisions with obstacles and other drones. Furthermore, the power consumed by the nodes in the considered test scenario, is reduced by 50% by utilising the proposed methodology.

Keywords: Autonomous swarm · Multi-agent systems · Agent-based modeling · Swarm intelligence · Leader follower

1 Introduction

Optimising different aspects in swarms of drones such as autonomous navigation, collision avoidance, payload reduction, resource allocation is gaining traction in the research community [1]. The deployment of swarms of UAVs adds remarkable advantages over single UAVs, as the UAVs in a swarm have the ability to work in a collaborative manner, and consequently have demand in various fields ranging from commercial use to search and rescue to military applications, and etc [2–4]. From the general perspective, the nodes or agents in the swarm can be classified

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

as: 1) reactive agents, where agents react to changes in the environment or a signal from another agent; 2) evolutionary agents, these are inspired by evolutionary algorithms and work on the basic principles of reproduction, mutation, recombination, and selection; 3) flocking agents, mimic the behaviour, inspired by e.g., swarms of bees or flocks of birds, moving together; 4) cognitive agents, inspired by the cognitive architecture, enables them to take decisions, process data, and make predictions [5–7].

Formation maintenance and collision avoidance are the most important problems in navigation of a swarm of drones [8,9]. In a formation the relative location of each agent is defined w.r.t. the other agents in the swarm, while the collision avoidance focuses on the path planning for individual agents and is responsible for avoiding any possible collisions between agents within the swarm and the agents and objects in the environment. The methodologies, for maintaining the formation, can be categorised into the following three generic approaches [10,11] : 1) virtual structure, in this approach the drones are collectively considered as a single drone and navigated through the trajectory as such [12–14]; 2) leader-follower, in this approach every drone functions individually and autonomously while maintaining and adjusting its position according to the leader and its neighbours in the formation [15–18]; 3) behavior based approach, where from a pre-defined strategy, one of the numerous behaviours is selected by the drone [19,20]. Furthermore, collision avoidance algorithms can be classified into the following three generic categories: 1) optimization based algorithms are focused on determining the near-optimal solutions for navigational and path planning purposes of each drone relative to other drones or objects in the vicinity, by relying on the already known locations and sizes of the statics objects for an efficient route calculation [21–23]; 2) sense and avoid based, by simplifying the process of collision avoidance to individual detection and avoidance, these have short response times and require much less computational power, individual drones are controlled without the knowledge of other drones paths etc [24–27]; and 3) force-field based, in this technique each drone is considered as a charged particle with attractive or repulsive forces. These attractive/repulsive forces between different drones themselves or drone and obstacles are used to generate the path for the drone to be taken [28,29].

In this paper, the leader-follower based approach is used for the swarm due to its ease of analysis, implementation, scalability, and reliability [18,24]. The focus is cost saving by reducing the processing power and reducing the payload carried by individual drones while still keeping the swarm autonomous and able to navigate to its destination. A global collision avoidance algorithm is defined for the leader agent in the formation. Which is then used by the follower agents in the adaptive autonomous mode to be able to calculate the relative locations/coordinates of the objects in the environment as observed by the leader, hence tackling the generic problem of tracking by a follower relative to the master.

The rest of the paper is structured as follows. Section 2 gives the motivation. Development of the proposed algorithm is described in Sect. 3. Section 4 focuses on simulation results. Finally, Sect. 5 concludes the paper with some discussion and future work.

2 Motivation

There are many scenarios when the intelligent decision making is not required to be done by every single agent in a swarm. Especially when it comes to static environments, not all drones in the swarm need to be equipped with all sensors and similarly all the drones are not supposed to be taking decisions intelligently. As it increases the processing power required by individual drones resulting in decreased battery timing and hence decreased mission life on one charge. For instance in huge warehouses or in cities with high rising buildings, where normally the dynamic variables w.r.t. obstacles approaches to zero, if only one drone i.e., the leader of the swarm, can see and do all the processing required to take the decisions, it takes a lot of processing load off the followers. In this paper we consider this technique to decrease the overall cost of the swarm, e.g., power consumption of sensors, by deliberately activating and deactivating some of the follower sensors in run-time. Taking inspiration from reactive agents [5] and using our proposed technique, for instance, if the leader detects any change in environment or dynamicity, it signals to the followers to turn on the adaptive autonomous high-conscious mode wherein the followers turn on more sensors to pass the critical situation, and later they switch back again to the low-conscious mode. In the case of the high-conscious mode, we utilised the collision avoidance technique in [24], which will be activated and used by individual drones of the swarm. The main motivation behind this is the cost reduction by reducing the power consumption and processing required by every individual drone in an autonomous swarm.

3 Proposed Approach

The general pseudo code of the global leader is given in Algorithm 1. We assume that the UAVs are assigned IDs and the leader-follower connection has established before the mission is started. Using the on-board processing units, each node is executing this top-level algorithm locally. Algorithm 1 starts by checking if the global leader for the swarm has been declared, and declares the leader if it has not been set up yet. After this, the followers are connected to their immediate leaders (Lines 3–8). After this, every node (starting from the leader) sends its coordinates, along with the distance and angle at which an obstacle (if any) has been detected, to its follower (Line 11–14). Based on the received signal and coordinates from *FollowerMode*, the node checks and sets the *Lead_is_Alive* flag depending on if its leader is still functioning properly or not (Line 14). If there's is no response from the leader, the node in question announces itself as its own leader. The node/leader cross-checks the coordinates it received from its follower(s) after the follower has calculated the distances and angles (Line 15). A criticality check is done by analysing if the absolute value of the received coordinates or angles is greater than a certain threshold, in this case the environment is defined as dynamic (Lines 16–18). If the node is itself the global leader or if the environment has been declared as dynamic, then the collision avoidance module is called (Line 20–21). Otherwise, if the node is not the global leader and the environment is not declared as dynamic, the *FollowerMode* module is called (Line 23–24).

In case the node itself is the leader, the timeout signal is not checked; otherwise, every node checks if it is receiving signals constantly from its respective leader. If the leader has not transmitted its coordinates by timeout, the follower declares itself as its leader and turns on its sensors for active collision avoidance maneuvering.

Algorithm 1 Global Leader

```
1: procedure NAVIGATION & OBJECT DETECTION
2:     $Dynamic \leftarrow False$;
3:     if $Self.ID == 1$ then
4:         $MyLeader \leftarrow Self$;
5:         $Lead\_is\_Alive \leftarrow False$;
6:     else
7:         $MyLeader \leftarrow Leader(Self)$;
8:         $Lead\_is\_Alive \leftarrow True$;
9:     end if
10:    while True do
11:        if $MyLeader == Self$ OR $Lead\_is\_Alive == False$ then
12:            $D_{obstacle}$, $A_{obstacle}$ ← Calculate obstacle distance and angles at which the edges lie;
13:        end if
14:        $Lead\_is\_Alive, ref.coords$ ← FollowerMode() ← send(
           $self.coord, angle, D_{obstacle}, Aobstacle$);
15:        $cal.ref.coords$ ← Reverse cross-check follower's received coordinates
16:        $CRITICALITYCHECK \leftarrow ref.coords - cal.ref.coords$
17:        if $|CRITICALITYCHECK| > Threshold$ then
18:            $Dynamic \leftarrow True$;
19:        end if
20:        if $Lead\_is\_Alive == False$ OR $Dynamic == True$ then
21:            $Dynamic$ ← AdaptiveAutonomousMode();
22:        else
23:            if $Lead\_is\_Alive == True$ AND $Dynamic == False$ then
24:                FollowerMode();
25:            end if
26:        end if
27:    end while
28: end procedure
```

3.1 Coordinate Calculation

In this function (specified in Algorithm 2), the follower receives the coordinates of the leader and the coordinates of the obstacle detected (if any). If there is no feedback from the leader by the timeout, the node returns false for its leader status (Lines 2–3). This information is then used by the node to declare itself as the leader for the fail safe mode in Algorithm 1 (Lines 21–22). Otherwise, based on its own coordinates, the node calculates the translational coordinates of the obstacle and returns the calculated values to be cross-checked (Lines 6–8), as shown in the Fig. 1 (Table 1).

Algorithm 2 FollowerMode

```
1: procedure FollowerMode($rcvcoord, rcvangle, dist.obstacle, angle.obstacle$)
2:     if $!rcvcoord$ AND $Timeout == 3$ then
3:         $Lead\_is\_Alive \leftarrow False$;
4:         return($Lead\_is\_Alive, self.coords$)
5:     else
6:         $Lead\_is\_Alive = True$;
7:         $self.coords \leftarrow$ Calculate new coordinates;
8:         return($Lead\_is\_Alive, self.coords$)
9:     end if
10: end procedure
```

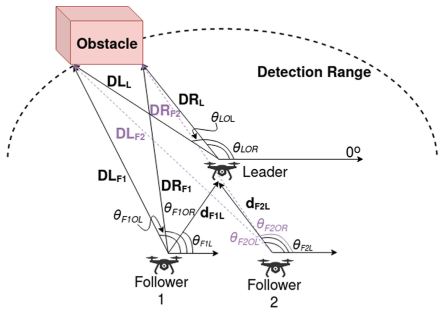


Fig. 1. Distance and direction calculation (Color figure online)

Table 1. Description of variables from Fig. 1

Variables	Description
DR_L DL_L	Distance of right and left edges of the obstacle from leader
DR_{F1} DL_{F1} DR_{F2} DL_{F2}	Calculated distance of obstacle's right and left edges from follower 1 and follower 2, respectively, as observed by leader
d_{F1L} d_{F2L}	Distance of leader from follower 1 and follower 2 respectively
θ_{LOR} θ_{LOL}	Angle at which right and left edges are detected from leader respectively
θ_{F1L} θ_{F2L}	Angle of leader from follower 1 and follower 2, respectively
θ_{F1OR} θ_{F1OL} θ_{F2OR} θ_{F2OL}	Angles at which right and left edges are detected from follower 1 and follower 2, respectively

3.2 Adaptive Autonomous Mode

If there are any obstacle(s) in the detection range, then this mode, Algorithm 3, is called by the leader. If the environment has been declared to be dynamic, then those node(s) which are working individually call this module locally. The system calls for collision avoidance module and keeps on checking the status of the environment (Line 2). In order to successfully avoid collisions, we utilise the collision avoidance technique presented in [24]. As soon as the collisions have been avoided, the status of the environment is set to static and the control is returned to the main module (Lines 3–5).

Algorithm 3 AdaptiveAutonomousMode

```
1: procedure AdaptiveAutonomousMode()
2:     Collision avoidance ($Dynamic$);
3:     if $!D_{obstacle}$ then
4:         $Dynamic \leftarrow False$;
5:         return($Dynamic$)
6:     end if
7: end procedure
```

4 Simulation Results

The initial conditions/assumptions for our work are defined as follows:

1. all UAVs have constant ground speeds
2. UAVs obtain their own position vectors using the on-board localization techniques
3. there is no information loss in communication between the UAVs

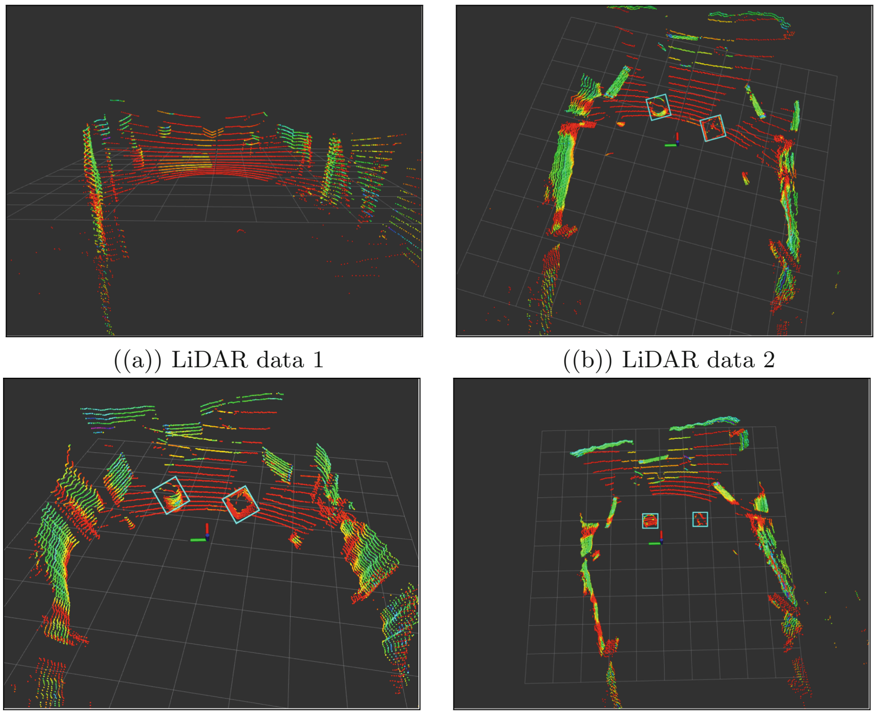
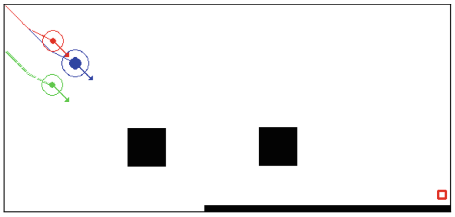

((a)) LiDAR data 1 ((b)) LiDAR data 2 ((c)) LiDAR data 3 ((d)) LiDAR data 4

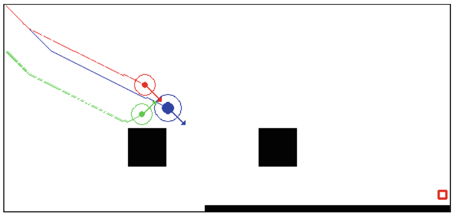
Fig. 2. LiDAR point set at starting point of simulation and when the obstacles are visible (Color figure online)

The LiDAR sensor used in our experiment is Velodyne Puck LITE. The data generated by Velodyne Puck LITE was then used and injected into the simulation platform for visualisation purposes, generation of the obstacle(s), and verification of the proposed algorithm.

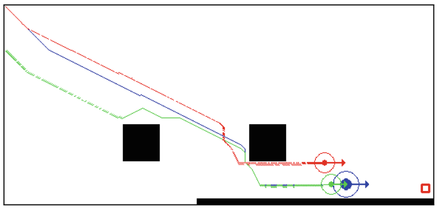
Figure 2, shows the LiDAR data from different angles and at different intervals for instance at starting point (Fig. 2(a)), when the obstacles enter detection range (Figure (b)), when the obstacles are in close vicinity (Figure (c) and 2(d)).



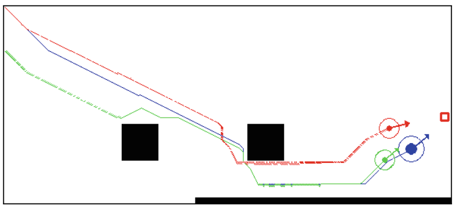
((a)) Drones start while maintaining formation, following the leader (in blue)



((b)) Obstacle comes in range, drone 3 turns on sensor for local collision avoidance



((c)) While going through obstacles



((d)) When there is no obstacle drone 3 comes back to initial formation shape

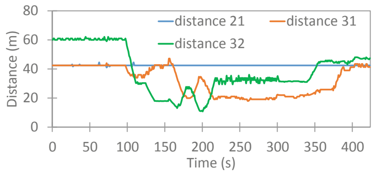
Fig. 3. Simulation Results at different instances (Color figure online)

In these figures, the position of the drone equipped with LiDAR is shown as the blue, red, and green bars indicating the z, y, and x-axes, respectively.

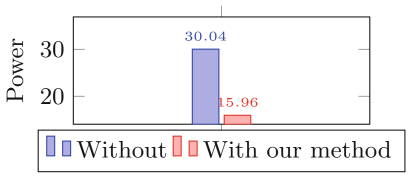
The simulation results shown in Fig. 3 show the V-shaped formation, in which UAV 1 (blue circle) is the leader (which is getting the data from LiDAR as shown in Fig. 2 and UAV 2 (red) and 3 (green) are followers calculating their coordinates as shown in Fig. 1. Figure 3(b) shows the scenario where the obstacle is within the detection range of the leader but not in its path. However, UAV3 upon performing necessary calculations realises that continuing along the current path will lead to a collision, and hence it deviates away from its original path as can be seen from the traces of the UAVs.

Figures 3(c) and 3(d) show the case when they go through the obstacles. Notice UAV2 does not deviate from its path and maintains the desired position w.r.t. UAV1 even when the second obstacle is close to it. That is because the calculations performed locally by UAV2 indicate that the collision is not possible given that the obstacle is stationary and out of the collision radius. However, after avoiding the second obstacle, UAV3 maintains the pre-defined minimum safe distance from UAV1 since going back to formation is not possible due to the third obstacle in the bottom. When the final destination is moved at run-time, Fig. 3(d), UAV3 comes back to its position in the formation, turns off the sensor, and starts following UAV1 by translating the coordinates transmitted by UAV1. Figure 4(a) shows the distances maintained by the UAVs throughout the simulation.

5000 mAh battery was used in the setup, and the power consumption of one Velodyne Puck LITE is 8 W (in typical conditions). Based on these values we calculated the power consumed by the sensors of all nodes, by tracking the



((a)) Distance maintained by each drone from its neighbours



((b)) Power Consumption by all nodes (mWh)

Fig. 4. Results

amount of time every sensor on each agent was active or turned on, during the simulation with and without our proposed methodology (shown in Figure (b)), where power consumption is given in milli-Watt hours. It is evident that utilising the proposed technique we can significantly reduce the power consumption of on-board sensors by approximately 50%, and hence increase the mission duration on one charge.

5 Conclusion

In this paper, we developed an algorithm for pre-defined formation of multi-agents in a static environment, where under normal circumstances one agent can see the surroundings and the other agents blindly follow. However, depending on the scenario the follower(s) can turn on their sensors for safety purposes. The proof of concept using simulation tools was performed to verify the method of translational coordinates calculation in a system consisting of multiple agents, i.e., a swarm of drones. The simulation results shown provide sufficient proof that the method works reliably in simulated static environments. It is evident from the results that the proposed algorithm helps reducing the power consumed by the sensors over time. In the considered test scenario it turned out to be approximately half the power consumed if the sensors are used in a continuous mode. In general, this power saving naturally depends on the dynamicity/structure of the environment. In future, we plan to further develop this algorithm by extending it for moving obstacles and test its usability in dynamic environments. This will be very interesting to analyse especially when two swarms have to cross paths i.e., multi-swarm cross-overs, as only the leader agents of both swarms can communicate, and the follower agents in the respective swarms use the leaders' translated coordinates.

References

1. Campion, M., Ranganathan, P., Faruque, S.: A review and future directions of UAV swarm communication architectures. In: 2018 IEEE International Conference on Electro/Information Technology (EIT), pp. 0903–0908, May 2018

2. Murray, R.: Recent research in cooperative control of multi-vehicle systems. J. Dyn. Syst. Meas. Control **129**, 571–598 (2007)
3. He, L., Bai, P., Liang, X., Zhang, J., Wang, W.: Feedback formation control of UAV swarm with multiple implicit leaders. Aerosp. Sci. Technol. **72**, 327–334 (2018). https://doi.org/10.1016/j.ast.2017.11.020, http://www.sciencedirect.com/science/article/pii/S1270963816309816
4. Ladd, G., Bland, G.: Non-military applications for small UAS platforms. In: AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference, p. 2046 (2009)
5. Mualla, Y., et al.: Agent-based simulation of unmanned aerial vehicles in civilian applications: a systematic literature review and research directions. Future Gener. Comput. Syst. **100**, 344–364 (2019). https://doi.org/10.1016/j.future.2019.04.051, http://www.sciencedirect.com/science/article/pii/S0167739X18328462
6. Gkiokas, A., Cristea, A.I.: Cognitive agents and machine learning by example: representation with conceptual graphs. Comput. Intell. **34**(2), 603–634 (2018). https://doi.org/10.1111/coin.12167, https://onlinelibrary.wiley.com/doi/abs/10.1111/coin.12167
7. Dorri, A., Kanhere, S.S., Jurdak, R.: Multi-agent systems: a survey. IEEE Access **6**, 28573–28593 (2018)
8. Zhuge, C., Cai, Y., Tang, Z.: A novel dynamic obstacle avoidance algorithm based on collision time histogram. Chin. J. Electron. **26**(3), 522–529 (2017)
9. Wang, X., Yadav, V., Balakrishnan, S.N.: Cooperative uav formation flying with obstacle/collision avoidance. IEEE Trans. Control Syst. Technol. **15**(4), 672–679 (2007)
10. Wei, R.: Consensus based formation control strategies for multi-vehicle systems. In: 2006 American Control Conference, pp. 6pp., June 2006
11. Low, C.B., Ng, Q.S.: A flexible virtual structure formation keeping control for fixed-wing UAVs. In: 2011 9th IEEE International Conference on Control and Automation (ICCA), pp. 621–626, December 2011
12. Beard, R.W., Lawton, J., Hadaegh, F.Y.: A coordination architecture for spacecraft formation control. IEEE Trans. Control Syst. Technol. **9**(6), 777–790 (2001)
13. Li, N.H., Liu, H.H.: Formation UAV flight control using virtual structure and motion synchronization. In: 2008 American Control Conference, pp. 1782–1787. IEEE (2008)
14. Dong, L., Chen, Y., Qu, X.: Formation control strategy for nonholonomic intelligent vehicles based on virtual structure and consensus approach. Procedia Eng. **137**, 415–424 (2016). Green Intelligent Transportation System and Safety
15. Oh, K.K., Park, M.C., Ahn, H.S.: A survey of multi-agent formation control. Automatica **53**, 424–440 (2015)
16. Buzogany, L., Pachter, M., D'azzo, J.: Automated control of aircraft in formation flight. In: Guidance, Navigation and Control Conference, p. 3852 (1993)
17. Shen, D., Sun, Z., Sun, W.: Leader-follower formation control without leader's velocity information. Sci. China Inf. Sci. **57**(9), 1–12 (2014)
18. Han, Q., Li, T., Sun, S., Villarrubia, G., de la Prieta, F.: "1-N" leader-follower formation control of multiple agents based on bearing-only observation. In: Demazeau, Y., Decker, K.S., Bajo Pérez, J., de la Prieta, F. (eds.) Advances in Practical Applications of Agents, Multi-Agent Systems, and Sustainability: The PAAMS Collection, pp. 120–130. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-18944-4_10
19. Lawton, J.R., Beard, R.W., Young, B.J.: A decentralized approach to formation maneuvers. IEEE Trans. Robot. Autom. **19**(6), 933–941 (2003)

20. Balch, T., Arkin, R.C.: Behavior-based formation control for multirobot teams. IEEE Trans. Robot. Autom. **14**(6), 926–939 (1998)
21. Zhang, X., Liniger, A., Borrelli, F.: Optimization-based collision avoidance. arXiv preprint arXiv:1711.03449 (2017)
22. Pham, H., Smolka, S.A., Stoller, S.D., Phan, D., Yang, J.: A survey on unmanned aerial vehicle collision avoidance systems. CoRR abs/1508.07723 (2015). http://arxiv.org/abs/1508.07723
23. Smith, N.E., Cobb, R., Pierce, S.J., Raska, V.: Optimal collision avoidance trajectories via direct orthogonal collocation for unmanned/remotely piloted aircraft sense and avoid operations. In: AIAA Guidance, Navigation, and Control Conference, p. 0966 (2014)
24. Yasin, J.N., Haghbayan, M.H., Heikkonen, J., Tenhunnen, H., Plosila, J.: Formation maintenance and collision avoidance in a swarm of drones. In: Proceedings of the 3rd International Symposium on Computer Science and Intelligent Control. ISCSIC 2019, Amsterdam, Netherlands. ACM, September 2019
25. Prats, X., Delgado, L., Ramirez, J., Royo, P., Pastor, E.: Requirements, issues, and challenges for sense and avoid in unmanned aircraft systems. J. Aircr. **49**(3), 677–687 (2012)
26. Albaker, B.M., Rahim, N.A.: A survey of collision avoidance approaches for unmanned aerial vehicles. In: 2009 International Conference for Technical Postgraduates (TECHPOS), pp. 1–7, December 2009. https://doi.org/10.1109/TECHPOS.2009.5412074
27. Soriano, A., Bernabeu, E.J., Valera, A., Vallés, M.: Multi-agent systems platform for mobile robots collision avoidance. In: Demazeau, Y., Ishida, T., Corchado, J.M., Bajo, J. (eds.) Advances on Practical Applications of Agents and Multi-Agent Systems, pp. 320–323. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38073-0_37
28. Albaker, B.M., Rahim, N.A.: Unmanned aircraft collision detection and resolution: concept and survey. In: 2010 5th IEEE Conference on Industrial Electronics and Applications, pp. 248–253, June 2010. https://doi.org/10.1109/ICIEA.2010.5516808
29. Seo, J., Kim, Y., Kim, S., Tsourdos, A.: Collision avoidance strategies for unmanned aerial vehicles in formation flight. IEEE Trans. Aerosp. Electron. Syst. **53**(6), 2718–2734 (2017)