2020 International Conference on Unmanned Aircraft Systems (ICUAS)
Athens, Greece. September 1-4, 2020

Obstacle Avoidance Manager for UAVs Swarm*

Ángel Madridano[1], Abdulla Al-Kaff[1], Pablo Flores[2], David Martín[1] and Arturo de la Escalera[1]

***Abstract*— Multi-Robot Systems for a swarm of autonomous UAVs are critical in providing essential solutions to numerous applications. However, working in complex environments requires higher levels of safety in navigation. One of the problems is the detection and avoidance of static and dynamic obstacles that appear in the UAVs paths.**

In this paper, two complementary methods are presented; to provide safe autonomous navigation of a swarm of UAVs. On the one hand, a method based on the Velocity Obstacle (VO) is responsible for avoiding collisions between the UAVs of the swarm by decreasing the velocity of the UAVs when they approach the same location. This method has the advantage of reducing the use of UAV resources and computational time. On the other hand, a method based on Light Detection and Ranging (LiDAR) information and Probabilistic Roadmaps (PRM) allows detecting dynamic obstacles appear in the path, exploring the environment in search of alternative paths, and finally establishing the one that minimizes the distance, and allows reaching a location avoiding the detected obstacles. The proposed methods have been tested and validated in simulation environments; as a previous step to their implementation in real UAVs. Moreover, the obtained results show the robustness and the efficiency of the system in detecting and avoiding the possible collisions.

I. INTRODUCTION

In recent years, the use of Multi-Robot Systems (MRS) have significantly grown; due to its versatility and robustness. The use of MRS with Unmanned Aerial Vehicles (UAVs), in particular, have been extended to numerous applications; such as search and rescue [1], [2], agriculture [3], [4], or communications networks [5], [6]. This growth is due to the facility of mounting sensors and actuators in the air, and because UAVs provide a powerful tool to perform dumb, dirty, and dangerous missions.

A key aspect of performing the tasks autonomously, is the ability to navigate in the environment safely, avoiding collisions with both static and dynamic obstacles. In UAVs swarm, beside the problem of detecting and avoiding obstacles in the environment, it is fundamental to locate and avoid collisions with the other UAVs in the swarm, as the scenario is shown in Figure 1, in which, the path planning algorithm considered the static obstacles while generating the trajectories [7]. However, the collision with dynamics obstacles still a problematic issue.

*This work is supported by the Comunidad de Madrid Government through the Industrial Doctorates Grants (GRANT IND2017/TIC-7834). We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used for this research.

The authors are with the Intelligent Systems Lab (LSI), Universidad Carlos III de Madrid, Avda. Universidad 30, 28911 Leganés, Madrid, Spain. {amadrida,akaff,dmgomez,escalera}@ing.uc3m.es

[2]The author is with Drone Hopper S.L., Avda. Gregorio Peces-Barba, 28919 Leganés, Madrid, Spain. pablo.flores@drone-hopper.com

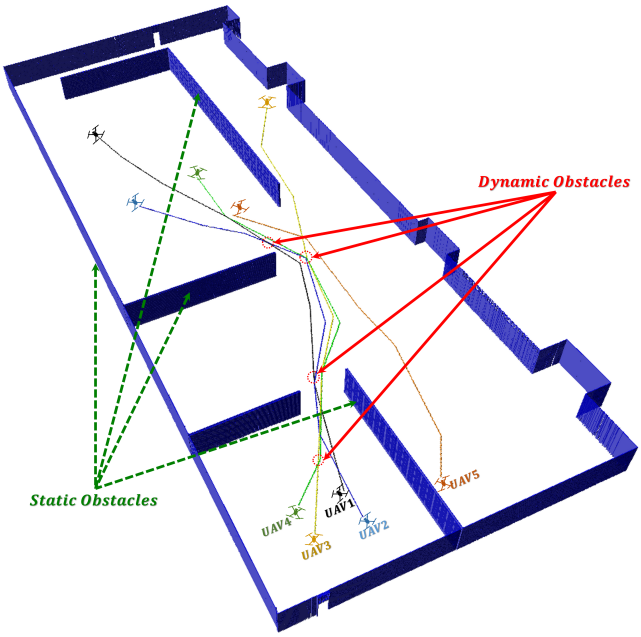


Fig. 1: Problem Statement.

This paper presents an architecture of providing each UAV in the swarm with an obstacle avoidance manager; for safe autonomous navigation. As it is shown in Figure 2, this obstacle avoidance manager is divided into two subsystems: The Velocity Control, which consists of a centralized architecture responsible for controlling the velocity of each UAV, to avoid collisions between them, based on the shared information about their positions. In this way, dynamic velocity profiles are established for each UAV, reducing its speed when several UAVs are approaching the same location. Thus, when the distance between two or more UAVs is below an empiric safety threshold, UAVs velocities are changed base on the reduction factor, preventing collision when each UAV passes that point at a different time. The second subsystem is the Obstacle Avoidance, which is a decentralized architecture based on LiDAR information, that allows the detection of obstacles in the environment and the re-planning of paths.

The remainder of this paper is organized as follows: Section II reviews the most recent works in the field of collision avoidance between agents of the same MRS. Section III describes the methods developed to avoid collisions between swarm UAVs, and individual UAVs with obstacles in the environment. Section IV collects and analyzes the results obtained from the implementation and simulation of both architectures. Finally, Section V summarizes the conclusions

978-1-7281-4277-7/20/$31.00 ©2020 IEEE

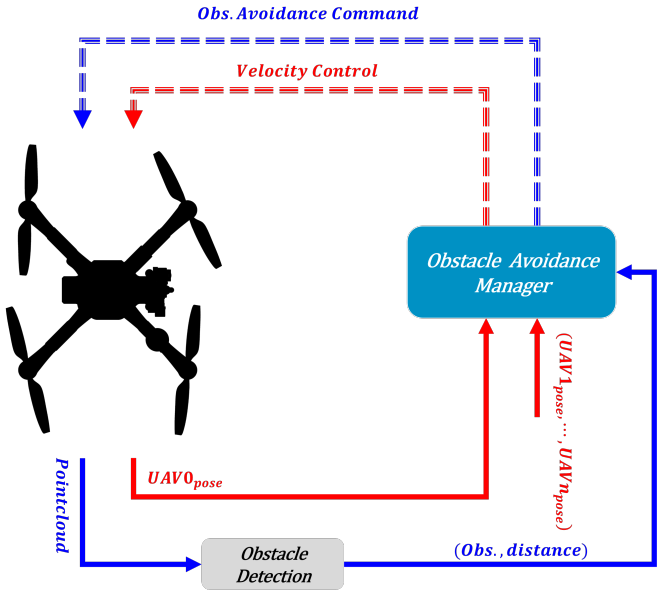


Fig. 2: Obstacle Avoidance Manager System.

and the future lines of research.

II. Related Works

The International Civil Aviation Organization (ICAO) is actively seeking technological solutions; to ensure the safety in the UAVs operations in shared airspace, and to provide these vehicles with capabilities similar to those of human-crewed aircraft through the pilot's senses [8].

From this aspect, different lines of research sought to provide a solution to deal with the problem of obstacles detection and avoidance; such as Artificial Potential Fields (APF) [9]–[11], Velocity Obstacles (VO) [12], [13], bio-inspired algorithms [14], [15], sampling-based methods [16]–[19], learning-based methods [20], and probability-based collision algorithms [21].

APF methods are based on repulsive fields to keep the UAVs away from obstacles, and thus, navigate through obstacle-free paths. In [10], a 3D mobility model based on APF method has been presented; to allow the UAVs to share information about the obstacles detected, and to provide a global 3D planning for all UAVs. The authors in [11], used APF-based method to obtain safe trajectories within a multi-UAV system, that flies in formation with a "Leader-follower" strategy. In addition, in [9], it has been introduced an optimized APF method for avoiding obstacles in a multi-UAV system that works collaboratively. Although these works presented novel solutions, applying APFs to multi-UAV systems, this technology leads to the problem to drive the vehicles to a local minimum, and then caused the difficulty to reach the desired final positions.

In VO, the aim is to predict the position, velocity, and acceleration of obstacles in the environment; to plan safe paths of the autonomous vehicle. In [12], a VO-based solution has been proposed for collision avoidance. In which, a combination of a cylinder protected zone and the VO pyramid is applied to avoid obstacles located in the UAV path. The results of this work showed the ability to avoidance the obstacles, however, this work focused only on detecting static obstacles. In [13], although there is no mention of a VO method, a solution based on a Kalman Filter is presented; to predict the position and movement of obstacles and thus, establish the maneuvers required to avoid them. The main disadvantage of this technology is the dependency on the vehicle's model, and the need to know precisely the movement of the obstacle.

Moreover, sampling-based methods can provide practical solutions; to deal with the problem of autonomous navigation. All these methods consist of low computational cost algorithms, that allows the re-planning of paths in a fast way. In [16], a LiDAR technology is used to create a safe path for a UAV; through Signed Distance Field (SDF) and Rapidly exploring Random Tree (RRT). A disadvantage of using this method for UAVs swarm is the need to create RRT for each individual UAV.

On the other hand, in [17] and [18], the authors used the disparity map from a stereo camera as an input. In [17], the information from the sensor is used to generate an occupation map; that provides information about which areas are free and which are occupied. While in [18], the sampling is done directly on the information coming from the sensor, that is, the disparity map is used to look for the free-collision areas, and to establish the necessary maneuvers to navigate through them. The disadvantage of these sensors is the presence of significant and uniform obstacles that affect the disparity and may not be detected.

Based on Deep Learning, Pinxin Long *et al.*, presented collision avoidance model, based on a decentralized framework [20]. In which, the data gathered from on-board sensors is used as an input for the training step of the reinforcement algorithm. This type of technology demonstrates the efficiency in achieving the obstacle avoidance, however, is requires a prior offline training. In addition, it requires high-performance on-board computers.

Finally, recently works [21] have appeared, in which within the global planning of the UAV swarm, the probability of collision with the rest of UAVs is calculated and, different moments of beginning are established to this way allow the safe navigation of all the UAVs.

This work aims to combine different strategies (layers 1, 2), to implement a system capable of allowing a swarm of UAVs to navigate safely through the environment. To achieve this, two layers of development are established:

- **First layer:** based on a centralized architecture, it focuses on avoiding collisions between the swarm's considering their positions over time, and acting on the velocity of UAVs to avoid collisions with each other, while navigating through previously defined paths.
- **Second layer:** based on a decentralized architecture, it focuses on detecting obstacles outside the swarms and establishing a path re-planning procedure, to avoid the collision.

III. Proposed System

This paper presents an obstacle avoidance manager for safe autonomous navigation of a swarm of UAVs, with the applicability to work in scenarios with or without GPS. As shown in Figure 2, the developed obstacle avoidance manager is divided into two layers; the first one is related to the Velocity Obstacle algorithms, in which the relative position of the UAVs is used to modify their velocity. While in the second layer, a sampling method based on Probabilistic Roadmaps (PRM) is used to establish dynamic paths, with the information from a LiDAR, to avoid collisions with objects detected in their paths.

A. Velocity Control

This first layer is applicable only on areas with available GPS, and it is based on establishing dynamic velocity profiles that allow safe movements of the swarm of UAVs in the work environment.

The architecture of this control is centralized, and a central node is responsible for receiving the GPS position of all UAVs and checking the distance between them. If the distance between two or more UAVs is less than the set threshold, the velocity of the UAVs is modified to avoid a possible collision. In this way, when two or more UAVs approach the same location, each one applies a different velocity profile; to pass at a different time to avoid the collision.

The velocity of the UAVs is modified using the reduction factor, which is set by the UAV's identifier. In this way, if more than two UAVs are below the safety threshold, they reduce their velocity in different proportions, the higher their identifier in the swarm, the greater the velocity reduction factor.

If the condition is met, this first UAV is a slowdown. Then the position of the second UAV is compared to the rest, and if the condition is met again, this UAV also slows down. This process is repeated until the last UAV with the highest identifier is reached, which is not compared to any other and therefore never slows down.

As shown in Algorithm 1, the first step starts with loading the UAV swarm configuration (identifier of each UAV and set of waypoints to travel). Once this step is completed, the algorithm enters a loop until all UAVs reach the target. During this loop, the waypoint to be reached is set in each UAV. At the same time, two situations are checked: if the waypoint has been reached and, the position of this UAV concerning the other UAVs, to apply the velocity reduction, in case it is at a distance lower than the safety threshold. The reduction factor is calculated as follows:

$$F_{Reduction} = (ID_{UAV} + 1) \times 3.0 \tag{1}$$

and the velocity of the UAV in the time instant $t+1$ is calculated directly:

$$Vel_{t+1}(UAV) = \frac{Vel_t(UAV)}{F} \tag{2}$$

The velocity reduction factor (Equation 1) is parameterized to differ for each UAV. Thus, when more than two UAVs must change their speed, they do so differently. Moreover, Equation 1 sets a multiplication factor that allows the difference in velocity between each UAV; to be large enough to avoid the collision. Finally, when the UAV reaches the target, it begins the necessary maneuvers until the landing is completed.

Algorithm 1: Velocity Controller

Input: *pose_sub* UAVs position, *agent_ID* Agent ID Vector, *agent_paths* Preset Path for each UAV.
Output: *velocity_pub* UAV Velocity.

```
Define: Wp_radius Range for reaching a waypoint, Safety_thr Safety distance between UAVs, agents Swarm agent vector, waypoints_i Vector with the waypoints of each path for each UAV, velocity_reducer Parameter in charge of reducing the nominal velocity, nWaypoint Current waypoint number
begin
    //Load swarm configuration
    for i = 0; i < agent_ID.size(); i++ do
        agents[i] = new Agent(agent_ID[i], agent_paths[i])
    waypoints_i ←generate_waypoints(agent_paths[i])
    velocity_reducer = 1
    //Velocity-based swarm control.
    for j = 0; j < agents.size(); j++ do
        //Calculate absolute distance to the next waypoint.
        Abs_Goal_distance_j = waypoints_j[nWaypoint_j] - pose_sub_j
        //Check the distance of the UAV from the other agents in the swarm.
        for h = j+1; h < agents.size(); h++ do
            euclidean ←distance_euclidean(pose_sub_j, pose_sub_h)
            if euclidean < Safety_distance then
                Velocity_reducer = (h+1)*3.0
        speed_pub_j = nominal_velocity / velocity_reducer
        //Check if the current waypoint has been reached.
        if Abs_Goal_distance_j < Wp_radius then
            if nWaypoint_j < waypoints_j.size() then
                //Load next Waypoint.
                nWaypoint_j = nWaypoint_j + 1
            else
                speed_pub_j = 0.0
```

B. Obstacle Detection.

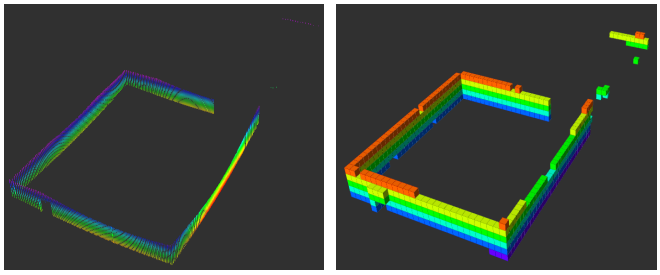
The velocity-based method is designed to allow UAVs to navigate safely, even if their preset paths coincide with other UAVs, reducing the use of the resources and the computational time. However, the centralized architecture presents the handicap that the loss of communications causes a lack of information among UAVs. For this reason, a LiDAR-based method is presented, in order to detect and avoid dynamic obstacles, that have not been previously mapped.

The objective of the second layer is to be able to detect obstacles external to the swarm, in order to avoid them, if necessary, by establishing an alternative path to the pre-established one.

To obtain the maximum amount of information from the environment, a LiDAR sensor capable of acquiring a large amount of data, in a range of 30 meters around the UAV, is mounted on the UAV.

The output of the LiDAR consists of a pointcloud (Figure 3a) that allows a 3D reconstruction of the environment. From this pointcloud, a 3D occupation map (Figure 3b) is generated, dividing the map into cells, and determining the state of these cells (occupied of free). The Octomap libraries [22] allows the implementation of the algorithm to take this pointcloud as an input and generates as an output a 3D occupation map of the environment on which to carry out the planning of new paths.

As the UAV navigates through the environment, the pointcloud and the occupancy map are updated. On this occupation map, a PRM-based path planning algorithm is executed. This PRM allows sampling of the environment in real-time and generates a set of possible paths so that the A* algorithm [23] is responsible for establishing the one that minimizes the distance traveled by the UAV between the initial and final points.



(a) LiDAR information (b) Occupancy Grid

Fig. 3: 3D Information

Figure 4 summarizes the obstacle detection and avoidance method. In which, the algorithm inputs are the current UAV position and the list of waypoints. Then, the UAV position and the goal waypoint are compared; to check if it has been reached. If not, it checks if the previous path is free-collision and moves the UAV in it if there are no obstacles (yellow line in Figure 5b), checking at each moment if the condition is still fulfilled.

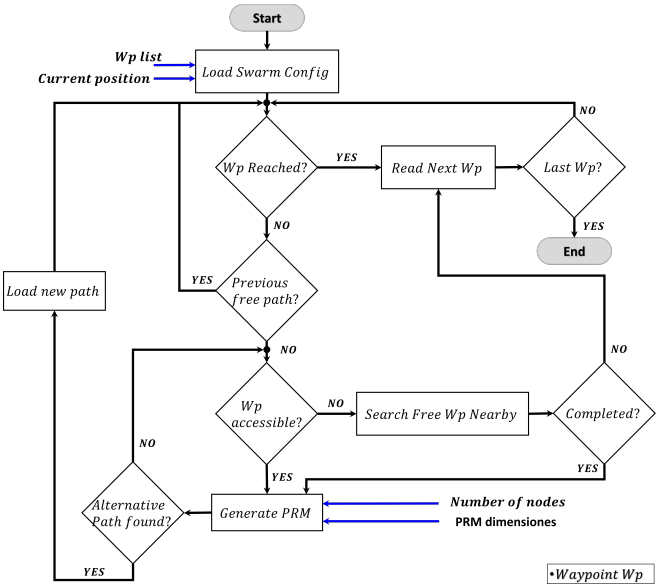
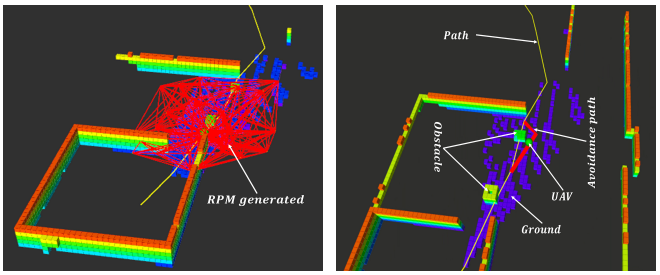


Fig. 4: Obstacle Avoidance Diagram

If an obstacle appears, the first step is to check if the waypoint is still accessible (not occupied by an obstacle). In this case, the algorithm searches for a free location as close as possible to the waypoint. If the it is not possible to find areas near the waypoint, the next waypoint is selected, and the process is repeated. If it is the last one, the mission is terminated.

In the case of an inaccessible waypoint, a new free-collision waypoint is generated, then the PRM is generated (Figure 5a), and the A* algorithm aims to find the best alternative path.



(a) Dynamic PRM Generation (b) New Generated Path

Fig. 5: Re-planning Path method

The PRM has as inputs a series of parameters that define it; such as the number of nodes to be generated, and the minimum and maximum dimensions of the Occupation Map. These dimensions are dynamically parametrizable depending on the distance of the waypoints. So that, if a waypoint is inaccessible and the next one is skipped, the area to be explored is necessarily larger. For this same reason, the number of nodes is also parameterized according to the dimensions, generating a greater number of nodes.

Once the PRM is generated, the A-Star algorithm is used to find the shortest path between the UAV and the target.

If there is no alternative path, the PRM will be generated again up to a maximum of 3 times, at which time it will be checked again if the waypoint is accessible.

On the other hand, if, on the contrary, an alternative solution is generated (red line in Figure 5b), the new waypoint is loaded, and the whole process described is repeated.

IV. Simulation & Results

The developed methods have been tested and validated with a simulated UAV software architecture of the real model SkyOnyx [24], observing their functionality as a previous step to their implementation in real platforms.

For all the tests carried out, Gazebo has been used as a simulation environment, and the Hector_Quadrotor ROS package is used for the simulation of dynamics and control of UAVs [25], [26].

In this work, the layers of the Obstacle Avoidance Manager have been simulated individually. On the one hand, a first scenario has been established, in which 5 UAVs navigate autonomously through previously established paths in a structured environment. This case allows validating the functioning of the velocity control algorithm for the UAV swarm [1]. On the other hand, the LiDAR-based algorithm has been tested in other more complex scenarios, with a higher density of obstacles, and without a previous mapping of that scenario [2].

As for the simulation of the obstacle avoidance method based on LiDAR, being a decentralized implementation, it has been tested with a single UAV. This UAV is equipped with a 16-plane LiDAR to re-plan its initial path according to the obstacles detected in the environment. The validation of this algorithm with the simulation of a single UAV is due to the need for high computational resources, to simulate multiple LiDARs on the same computer. Even so, the environment has included models of UAVs performing hovering maneuvers, to test that the algorithm is not only capable of detecting large obstacles, but that the LiDAR can collect information from other UAVs present in the route of the UAV, and be able to perform an avoidance maneuver.

The loss of communications and the presence of UAVs outside the swarm are two situations that can happen in real maneuvers. For this reason, this second layer within the Obstacle Avoidance Manager allows the action to be taken in such cases.

Furthermore, in this method, if the predefined path is modified if an obstacle external to the swarm is detected. In the case of the velocity-based method, the velocity of the UAVs is acted upon in order to complete the preset paths without colliding between the members of the swarm.

Figures 6, 7, 8 and 9 shows the results of the obstacle velocity algorithm, where the relative distance of one UAV to the others in the swarm is shown. From these figures, it is illustrated that when the distance is below the safety threshold ($3.5m$), the speed reduction factor increases, slowing down the UAV; to avoid the collision.

[1] https://youtu.be/_PU_Q9vHtzU
[2] https://youtu.be/Q254g5lr_5A

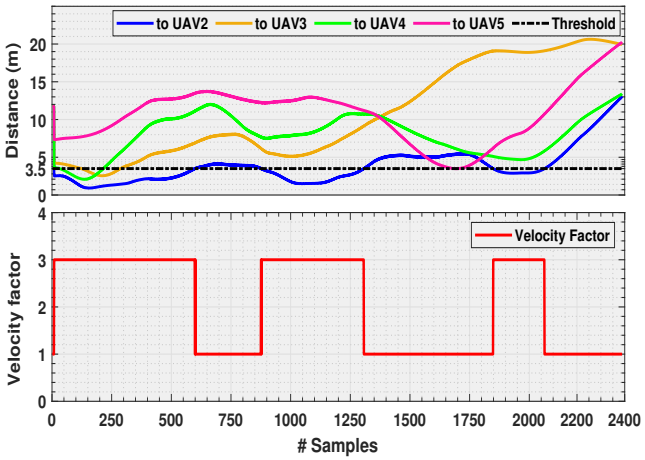


Fig. 6: Relative distance vs. Velocity reduction factor UAV_1.

Figure 6 compares the Euclidean distance of the first UAV (UAV_1) to the others UAVs in the swarm, and how its speed changes accordingly with the approaching to the other UAVs. It is shown that in three instants, UAV_1 reduces its speed by a factor of 3. In the first instant, this reduction is due to the proximity of 3 UAVs (UAV_2, UAV_3, and UAV_4), while in the two following instants, the proximity occurred with UAV_2.

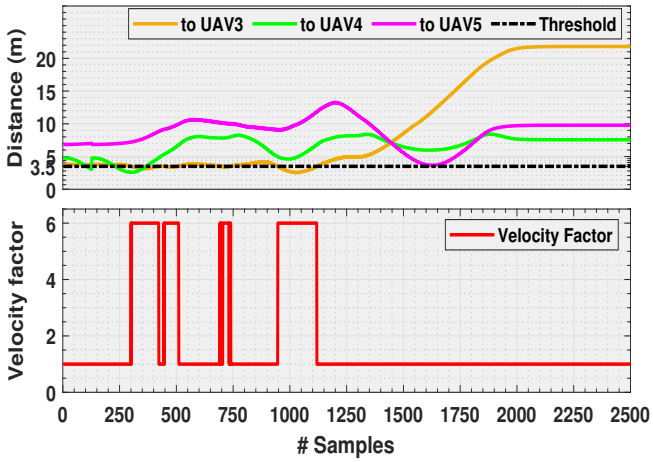


Fig. 7: Relative distance vs. Velocity reduction factor UAV_2.

Figure 7 shows the relative distance of UAV_2 and the other UAVs. In this case, UAV_1 with the lowest identifier is not included; because it has been compared in Figure 6, and has already reduced its velocity. Therefore, each UAV is compared with the UAVs with higher identifiers, and so on (Figures 8 and 9).

In addition, it is also noted from Figures 6, 7, 8 and 9 that the velocity factor of each UAV increases with respect to the UAV identifier; $F = 3$ in the case of UAV_1, $F = 6$ for UAV_2, and $F = 9$ and $F = 12$ for UAV_3 and UAV_4 respectively. These increment was considered because if the UAV identifier, means higher the UAV velocity, and it required greater value of the factor.

Finally, there are no results shown for UAV_5 since it is the last agent of the swarm and has already been compared

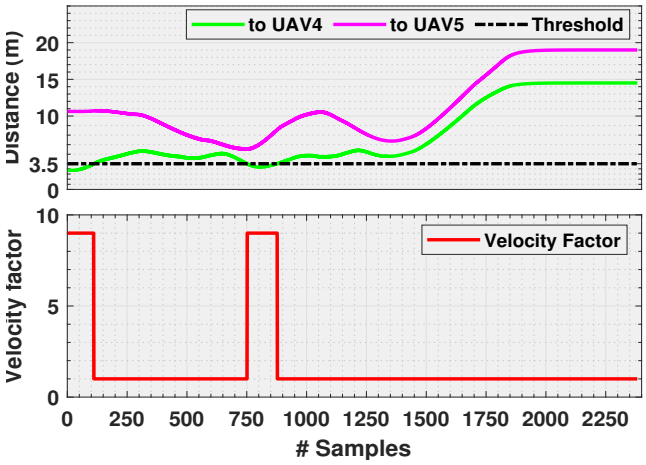


Fig. 8: Relative distance vs. Velocity reduction factor UAV_3.

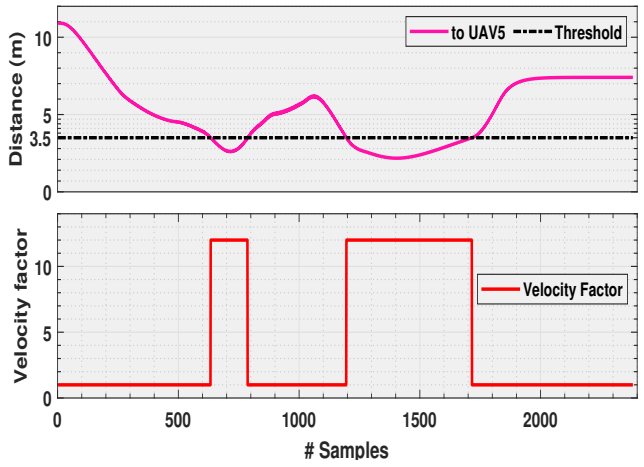


Fig. 9: Relative distance vs. Velocity reduction factor UAV_4.

with the rest of the UAVs. Therefore, this last agent always moves at its nominal velocity and is the one with the highest priority in the swarm.

V. Conclusions & Future Works

This paper presents two methods that allow a swarm of autonomous UAVs to navigate safely, with the ability to detect and avoid static and dynamic obstacles. First, a method based on VO is implemented, in which, by knowing the UAVs position at each instant of time, velocity reductions are applied to the UAVs to avoid possible collisions. In this way, based on previous safe trajectory planning, a swarm of UAVs can be launched to navigate in a common area safely, and avoid collisions between them.

On the other hand, a method based on LiDAR is presented to the detection and avoidance of dynamic obstacles. This method complements the previous one, and allows the detection of obstacles, both dynamic and static, different from the UAVs that are part of the swarm. Furthermore, it allows detecting and avoiding the UAVS of the swarm in case of losing the communications between them.

The future works will focus on implementing an adaptive velocity reduction, so that the reduction value is not fixed for each UAV, but can be adapted, and increased or decreased according to the speed of the obstacle and the distance to it. In addition, implement and test the methods with real UAVs. Once this step has been achieved, their possible use in different types of autonomous vehicles and their applicability to problems; such as full coverage are another field in which to continue this work.

References

[1] E. T. Alotaibi, S. S. Alqefari, and A. Koubaa, "Lsar: Multi-uav collaboration for search and rescue missions," *IEEE Access*, vol. 7, pp. 55 817–55 832, 2019.

[2] H. Kurdi, J. How, and G. Bautista, "Bio-inspired algorithm for task allocation in multi-uav search and rescue missions," in *Aiaa guidance, navigation, and control conference*, 2016, p. 1377.

[3] T. Elmokadem, "Distributed coverage control of quadrotor multi-uav systems for precision agriculture," *IFAC-PapersOnLine*, vol. 52, no. 30, pp. 251–256, 2019.

[4] C. Ju and H. I. Son, "Multiple uav systems for agricultural applications: control, implementation, and evaluation," *Electronics*, vol. 7, no. 9, p. 162, 2018.

[5] P. Chandhar, D. Danev, and E. G. Larsson, "Massive mimo for communications with drone swarms," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1604–1629, 2017.

[6] A. A. Khuwaja, Y. Chen, N. Zhao, M.-S. Alouini, and P. Dobbins, "A survey of channel modeling for uav communications," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2804–2821, 2018.

[7] Á. Madridano, A. Al-Kaff, D. Martín *et al.*, "3d trajectory planning method for uavs swarm in building emergencies," *Sensors*, vol. 20, no. 3, p. 642, 2020.

[8] A. Maccapani and M. Goiak, "Detect and avoid: an industrial perspective technical and certification considerations for automatisms and safety functions associated with unmanned aircraft systems," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018, pp. 68–73.

[9] J. Sun, J. Tang, and S. Lao, "Collision avoidance for cooperative uavs with optimized artificial potential field algorithm," *IEEE Access*, vol. 5, pp. 18 382–18 390, 2017.

[10] E. Falomir, S. Chaumette, and G. Guerrini, "A 3d mobility model for autonomous swarms of collaborative uavs," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 196–204.

[11] J. Zhang, J. Yan, X. Xing, D. Yuan, X. Hou, X. Qi, and P. Zhang, "The collision avoidance control algorithm of the uav formation flight," in *Proc. ARAA*, 2017, pp. 1–7.

[12] C. Y. Tan, S. Huang, K. K. Tan, and R. S. H. Teo, "Three-dimensional collision avoidance design on unmanned aerial vehicle," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018, pp. 521–530.

[13] M. Ille and T. Namerikawa, "Collision avoidance between multi-uav-systems considering formation control using mpc," in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2017, pp. 651–656.

[14] J. Zhao, C. Hu, C. Zhang, Z. Wang, and S. Yue, "A bio-inspired collision detector for small quadcopter," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–7.

[15] A. Al-Kaff, F. García, D. Martín, A. De La Escalera, and J. M. Armingol, "Obstacle detection and avoidance system based on monocular camera and size expansion algorithm for uavs," *Sensors*, vol. 17, no. 5, p. 1061, 2017.

[16] L. Lu, C. Sampedro, J. Rodriguez-Vazquez, and P. Campoy, "Laser-based collision avoidance and reactive navigation using rrt* and signed distance field for multirotor uavs," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 1209–1217.

[17] C. Lim, B. Li, E. M. Ng, X. Liu, and K. H. Low, "Three-dimensional (3d) dynamic obstacle perception in a detect-and-avoid framework for unmanned aerial vehicles," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 996–1004.

[18] E. Perez, A. Winger, A. Tran, C. Garcia-Paredes, N. Run, N. Keti, S. Bhandari, and A. Raheja, "Autonomous collision avoidance system for a multicopter using stereoscopic vision," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018, pp. 579–588.

[19] J. Hu, M. Wang, C. Zhao, Q. Pan, and C. Du, "Formation control and collision avoidance for multi-uav systems based on voronoi partition," *Science China Technological Sciences*, vol. 63, no. 1, pp. 65–72, 2020.

[20] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6252–6259.

[21] W.-H. Ko and P. Kumar, "Probability-based collision detection and resolution of planned trajectories for unmanned aircraft system traffic management," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 946–951.

[22] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com

[23] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[24] A. Al-Kaff, R. Alonso, M. Osman, and A. Hussein, "Skyonyx: Autonomous uav research platform for air transportation system (atsys)," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3550–3555.

[25] J. Meyer, "Hector quadrotor ros package website," *www., Available: http://wiki. ros. org/hector_quadrotor*, 2014.

[26] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.