A Centralized Control of Movements Using a Collision Avoidance Algorithm for a Swarm of Autonomous Agents

Kleber Loayza, Pedro Lucas and Enrique Peláez
ESPOL Polytechnic University, Escuela Superior Politécnica del Litoral, ESPOL,
Electrical and Computer Engineering Department, Centro de Tecnologías de Información,
Campus Gustavo Galindo Km.30.5 Vía Perimetral, P.O. Box 09-01-5863,
Guayaquil, Ecuador
Email: {kloayza, pedro.lucas, epelaez}@cti.espol.edu.ec

***Abstract*—This work proposes a method for moving a swarm of autonomous Unmanned Aerial Vehicles to accomplish an specific task. The approach uses a centralized strategy which considers a trajectory calculation and collision avoidance. The solution was implemented in a simulated scenario as well as in a real controlled environment using a swarm of nano drones, together with a setup supported by a motion capture system. The solution was tested while planting virtual seeds in a field composed by a grid of points that represent the places to be sown. Experiments were performed for measuring completion times and attempts to prevent impacts in order to test the effectiveness, scalability and stability of the solution as well as the robustness of the collision avoidance algorithm while increasing the number of agents to perform the task.**

***Keywords*—agents behavior, swarm systems, collision avoidance, farming automation, UAV**

I. Introduction

During the last couple of years, the use of *UAVs (Unmanned Aerial Vehicles)* in different domain applications has increased, in particular to perform complex tasks; such domain applications are taking advantage of the flexibility and robustness that swarm behavior can provide, where a group of individuals agents perform simple tasks, but collaboratively can overcome the complexity of hard to solve problems [1].

One application for a swarm of these agents is related to precision agriculture, where the accomplishment of specific tasks, for responding to inter and intra-field variability in crops, can provide the means for optimizing costs and returns on investments while preserving resources [2].

The swarm technique developed in this research has been tested in sowing seeds, a task in precision agriculture which demands scalability and stability. Applications like this task can be automated with a swarm of autonomous individuals, which requires methods to coordinate agents and distribute activities considering movements and collision avoidance. There is a need to develop these methods, as the related work indicates, for the mentioned task and others.

The contribution of this work is related to trajectories, in which the solution involves movement control to distribute a group of agents across a field to be virtually seeded, as well as a collision avoidance algorithm among the agents. This research is considering a centralized model to control and provide feedback in real time to the agents. Our goal is to develop a model, based on a swarm of simple UAVs, to coordinate agents and distribute activities to them for accomplishing an specific task; also, test its effectiveness, scalability and stability as well as the robustness of the collision avoidance algorithm with regard to possible impacts while increasing the number of individuals.

A. Related Work

There have been different applications for using flying agents for precision agriculture [3], applications which have been focused on vision processing to map areas for improving farming activities such as pest detection, fumigation and soil analysis. Other applications are focused on planting advice to determine variable planting rates to accommodate varying conditions across a single field, in order to maximize yield [4]. In other cases, a combination of several elements has been considered, as the strategy for trajectories to follow proposed by Costa *et al* [5], in which the algorithm is used to disperse chemicals over a ground, and control a simulated swarm of agents in a modeled climate condition.

Considering movements in this kind of applications, flying agents face coordination problems for adopting suitable group formations and collision avoidance. Meng *et al* in [6] proposed a synergy of *Ant Colony Optimization (ACO)* and *Particle Swarm Optimization (PSO)* algorithms for converging to a target and for avoiding collisions with obstacles and among themselves. Zhou *et al* [7] developed a user-operated quad-rotor swarm which avoids collisions and maintains a desired formation by using a virtual rigid body that groups the agents, and a vector field that repels other quad-rotors, a similar approach was used by Kim *et al* [8] who proposed a framework for a swarm system based on *Artificial Potential Functions (APF)* for attractive and repulsive forces defined by vector fields. Sharma *et al* in [9] also proposed a solution for

978-1-5386-3894-1/17/$31.00 ©2017 IEEE

the collision problem by using swarming and motion laws in a detailed mathematical description.

Some researchers have also tested their strategies to solve these coordination problems in simulations [5] [6], others with real flying agents or both [7] [10]. This work proposes a model based on a centralized movement control to provide feedback in real time, as well as a collision avoidance algorithm. The model has been tested in a simulated scenario as well as in a real controlled environment.

II. Background

A. Swarm Intelligence

Inspired by nature, *Swarm Intelligence* (SI) can be defined as the collective behavior of decentralized and self-organized artificial individuals [11]. Such individuals show limited abilities to solve complicated tasks; but, when they are organized in groups, a complex behavior emerges to complete those tasks easily using local communication and information transmission for mutual cooperation [12].

According to Meng *et al* [6], there are features that SI emphasizes when solving a problem: *self-organization*, *parallelism*, and *exploitation of direct (peer-to-peer) or indirect (via the environment) local communication mechanisms among agents*. Tan Ying *et al* [12] add other characteristics that support the SI approach, such as: *scalability*, *stability*, *economical*, and *energy efficiency*. These descriptors allow to build systems with great flexibility and robustness.

This work is focused on a swarm of *UAVs (Unmanned Aerial Vehicles)* which are supported by a centralized configuration through an indirect local communication mechanism, as described in section III. A centralized approach shows low variability in task accomplishment and reaches a solution with similar performance as compared to distributed strategies [10].

B. UAVs in Agriculture

UAVs, known as drones, have an increased number of applications thanks to the robust and sustained investment, as well as to the relaxed regulatory environment, which have become appealing to entrepreneurs for creating startups that provide services for drone-planting systems in precision agriculture as Mazur [4] claims.

Reforestation can be performed now using drones that map potential seeding fields, then automated sowing drones carrying seed pods would fly above the ground, following a pre-determined planting pattern, and fire germinated seeds into the soil [13] [14]. Currently, this process is possible with manual piloted drones or automated agents, hence a solution that involves a swarm of autonomous UAVs is a field of interest in this domain [4].

III. Methodology

The strategy for accomplishing the task of sowing was intended to be executed in a controlled environment using reals UAVs. For the experiments, Crazyflie 2.0 nano drones [15] were used as the agents to form a swarm. This setup is composed by drones, a *Motion Capture System (MOCAP)* and two workstations, one for controlling the agents and the other for managing the MOCAP. These systems allow the drones to get stable in an specific point in space, through the feedback provided by the MOCAP's system which communicates, via a local network, with the controlled stations, as well as with the firmware on each agent. To estimate a position and be able to travel from point to point in a real environment, the system uses an *Extended Kalman Filter* as in [16]. This setup is shown in Fig. 1.

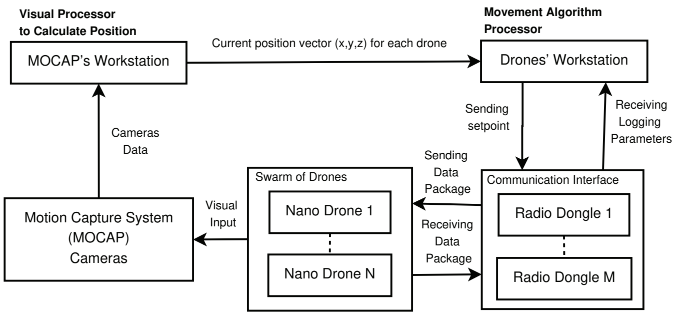

Fig. 1: Architecture for controlling a swarm of nano drones

In order to test the effectiveness of the centralized model to control and provide feedback in real time to the agents, as well as the collision avoidance algorithm while increasing the number of agents to perform the task of sowing seeds, the solution was implemented first in a simulation environment provided by the game engine Unity3D [17]. The centralized algorithms were run in a simulated field with a grid of 10 x 10 holes, which will be intended to be virtually seeded by the swarm of drones. There were 30 experiments performed using 1 to 30 agents in each trail; that is, up to 900 tests. Also, static obstacles were distributed randomly across the field. The agents have the same configuration and share the same set of features, such as speed, height while they are moving, height while they are planting, size, and collision radius. After each run, the time for accomplishing the task was taken for each amount of drones.

Once the simulation showed the drones are not affected by collisions or unexpected behaviors, the solution was implemented in a controlled environment as described before. The tests included the centralized communication strategy, number of drones and collision avoidance with groups from 1 to 5, and 30 experiments per trail. These experiments followed the same guidelines as the simulated solution.

IV. Centralized Swarm Method

To perform the task defined as sowing seeds, the swarm of drones will have to resolve the needed movements for planting in a ground represented as Pg and modeled as a grid composed by r rows and c columns, where each element is a position in space defined by equation (1).

$$Pg = \{(x, y, z) : x, y, z \in \mathbb{R}, y = 0\} \tag{1}$$

Each point in Pg is separated by a distance sep from each other. Pg is treated as a one-dimensional array which

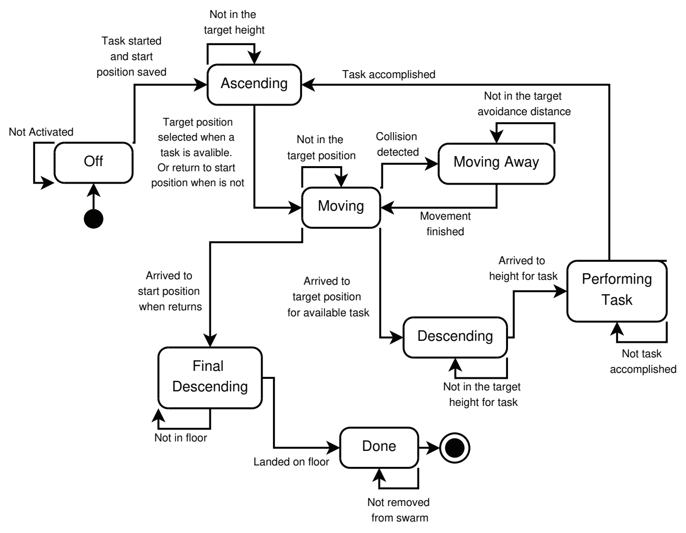
Not in the target height
Task started and start position saved
Ascending
Task accomplished
Not in the target avoidance distance
Not Activated
Off
Target position selected when a task is avalible. Or return to start position when is not
Not in the target position
Collision detected
Moving Away
Moving
Movement finished
Arrived to height for task
Performing Task
Arrived to start position when returns
Arrived to target position for available task
Descending
Not task accomplished
Final Descending
Not in the target height for task
Not in floor
Done
Landed on floor
Not removed from swarm

Fig. 2: Finite state machine of an agent

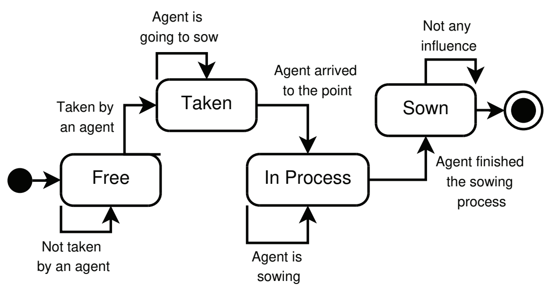

Fig. 3: Finite state machine for sowing a seed

is traversed by the agents to select the position where they need to go in order to collaborate in the seeding task.

An agent A behaves in real time according to a *finite state machine*, as described in Fig. 2. As it is shown, an agent starts ascending to a defined height h_m, starts to move to a selected point, moves away from a near agent for avoiding collisions, arrives to its goal, descends to an specific height h_p near the ground, performs the task of planting the seed in an available hole in a time t_p, ascends, and moves to the next hole available to be sowed. This process is performed until all the holes are filled, which is managed by a central controller that changes the states for each hole depending on the actions carried out by the agents. The states for the hole on the ground are defined as in Fig. 3.

An agent selects the place to go by traversing the set of points Pg and finding the nearest distance between its current position $\vec{pa}$ and the next goal by using the *Euclidean Distance* for vectors defined by equation (2). This selection is carried out after the agent finishes ascending, and depending on the availability of free holes to seed, the agent moves to the nearest point or return to its starting point if there is not available holes to be seeded. The behavior of the agents is individual, but they share a centralized parameter, the set of points Pg, which tells them where to move next.

$$D_{ap} = ||\vec{Pg_i} - \vec{pa}|| : \vec{pa}, \vec{Pg_i} \in \mathbb{R}^3, i \in [0, r \cdot c] \qquad (2)$$

The ascending and descending behaviors consider a speed s_v for performing a displacement to h_m or h_p for a period of time Δt given by equation (3), which uses the unit vector $\hat{\mathbf{j}}$ to move vertically. This equation takes into account an ascending movement and for descending, the unit vector $\hat{\mathbf{j}}$ is inverted.

$$\vec{pa} = \vec{pa} + s_v \Delta t \hat{\mathbf{j}} : \vec{pa}, \hat{\mathbf{j}} \in \mathbb{R}^3, s_v \in \mathbb{R} \qquad (3)$$

The strategy, for moving to an available hole, leads an agent throughout a straight line, from its current position $\vec{pa}$ to a target position $\vec{pt}$ with a speed s_m. The target position is a selected position $\vec{Pg_i}$ in the set of points of the space Pg, or the starting point $\vec{pa}_{init}$ when the agent finishes its task and returns. The direction of movement is given by the unit vector defined in (4) which is used in the movement equation (5) that allows the agent to advance for a time Δt until it arrives at the target $\vec{pt}$.

$$\hat{\mathbf{dir}} = \frac{\vec{pt} - \vec{pa}}{||\vec{pt} - \vec{pa}||} : \vec{pa}, \vec{pa} \in \mathbb{R}^3 \qquad (4)$$

$$\vec{pa} = \vec{pa} + s_m \Delta t \hat{\mathbf{dir}} : \vec{pa}, \hat{\mathbf{dir}} \in \mathbb{R}^3, s_m \in \mathbb{R} \qquad (5)$$

Given that a group of agents in selected positions might try to concentrate, provoking potential collisions among them, a collision avoidance algorithm has been designed. The algorithm is monitored by a centralized controller, which provides the position for all agents and verifies how close they are to each other depending on a radius r_c. The collision avoidance strategy is defined in the Algorithm 1 below, which uses the *Euclidean Distance* for checking the closeness among agents and selects the directional vector $\hat{\mathbf{coll}}$ by determining the nearest element that is going to collide.

When a possible collision is detected by the $isNear$ indicator, the agent changes its direction of traveling, opposite to the collision, by using the normalized vector $\hat{\mathbf{coll}}$, as defined in equation (6) below. The directional vector $\hat{\mathbf{av}}$ to avoid the collision is defined as:

$$\hat{\mathbf{av}} = \frac{\vec{coll}^{\perp} - \vec{coll}}{||\vec{coll}^{\perp} - \vec{coll}||} : \vec{coll} \in \mathbb{R}^3 \qquad (6)$$

The collision detection forces the agent to change its behavior to a *Moving Away* state in order to move a distance d_c away from the other agents by using the directional vector defined in equation (6). To achieve this displacement, the agent uses equation (8) for moving away during a time t_c as defined by (7). Once the agent is not in the collision range, it retakes the trajectory to the target; that is, returns to *Moving* state.

$$t_c = \frac{d_c}{s_m} : d_c, s_m \in \mathbb{R} \qquad (7)$$

$$\vec{pa} = \vec{pa} + s_m \Delta t \hat{\mathbf{av}} : \vec{pa}, \hat{\mathbf{av}} \in \mathbb{R}^3, s_m \in \mathbb{R} \qquad (8)$$

```
Data: currentAgent, AllAgents[]
Result: isNear, $\hat{\mathbf{coll}}$
minimalDistance := +infinity;
$\hat{\mathbf{coll2D}}$ := (0,0);
isNear := False;
for each otherAgent in AllAgents do
    if currentAgent is not otherAgent and
      otherAgent.state is not Off and otherAgent.state is
      not Done then
        $\overrightarrow{otherAgentPosition2D}$ :=
          (otherAgent.position.x, otherAgent.position.z);
        $\overrightarrow{currentAgentPosition2D}$ :=
          (currentAgent.position.x,
          currentAgent.position.z);
        $\overrightarrow{displacement2D}$ := $\overrightarrow{otherAgentPosition2D}$ -
          $\overrightarrow{currentAgentPosition2D}$;
        distance := $||\overrightarrow{displacement2D}||$;
        if distance is less or equals to
          currentAgent.collisionRadius then
            if distance is less than minimalDistance then
                minimalDistance := distance;
                $\hat{\mathbf{coll2D}}$ := $\frac{\overrightarrow{displacement2D}}{||\overrightarrow{displacement2D}||}$;
            end
            isNear := True;
        end
    end
end
$\hat{\mathbf{coll}}$ := ($\hat{\mathbf{coll2D}}$.x, 0, $\hat{\mathbf{coll2D}}$.y);
```

Algorithm 1: Collision avoidance algorithm

V. Results

Two environments were used for testing the proposed solution; a simulation and a real setup. The simulation was run for groups of individuals from 1 to 30 with 30 experiments for each group, in a field configured as a grid of $10x10$ with 20 obstacles placed randomly. The simulator *Unity3D* uses a programming paradigm based on *components*, so the agents are objects with an individual behavior which is implemented as one of its components that considers the finite state machine in Fig. 2 from section III, as well as the ground, which is another object that is manage by the centralized control. As shown in Fig. 4 the agents, illustrated as circles, are spread out to solve the task on the field at different times for a swarm of 30 drones. The holes are represented by squares along the ground and the obstacles by diamonds.

In the real setup, the field has been configured with a size of $4x4$ because of space limitation and MOCAP's effectiveness. The agents were organized in groups of drones from 1 to 5 with 30 experiments for each group and 3 obstacles distributed randomly. Fig. 5 shows a picture of the real setting and the swarm of drones performing the task, where the grid of holes for seeding can be identified as white spaces on the floor, the drones are circled, and the obstacles are enclosed in a square.

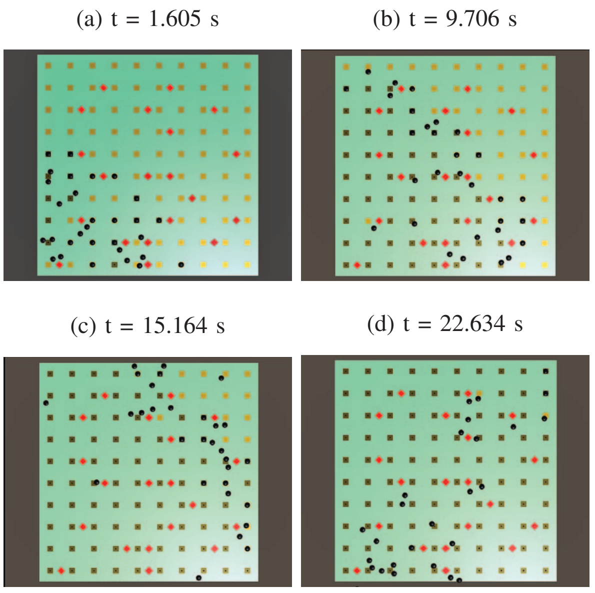

Fig. 4: Simulation screen-shots for different times considering a group of 30 drones

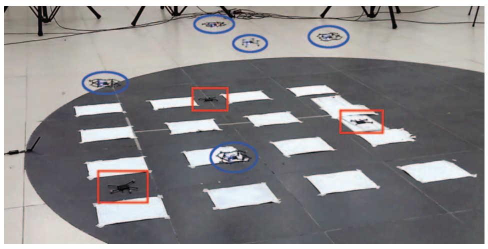

Fig. 5: A group of 5 nano drones performing planting movements

TABLE I: Non-linear regression results for time

Parameter	Estimate	Std. Error	t value	Pr($>\|t\|$)
a	404.52548	3.28133	123.28	<2e-16
b	0.24733	0.01282	19.29	<2e-16

The obstacles are placed on the floor level because they are tracked by the MOCAP like the other drones but static, and considered by the system as elements of infinite height.

For the simulation, the planting time for each swarm of drones was registered and compared among themselves to determine how this time improved as the number of agents increased; a relationship between the number of drones and the time spent to finishing the task was established as the function $y = \frac{a}{x+b}$. A non-linear regression with an estimation of goodness of 0.9944 was also established, as described in Table I. Fig. 6 shows this behavior below.

The number of drones is compared against the number of collision avoidances in simulation. In this case, the behavior is according to a linear relationship as Fig. 7 describes; that is, a function $y = ax + b$. A simple linear regression with an estimation of goodness of 0.9764 was calculated to establish

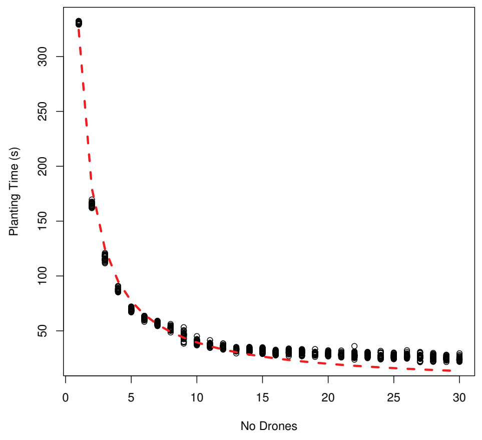

Fig. 6: Curve: Number of Drones vs Planting Time

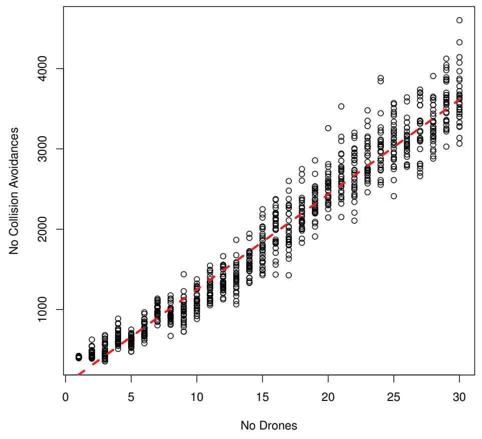

Fig. 7: Curve: Number of Drones vs Number of Collision Avoidances

TABLE II: Linear regression results for collision avoidances

Parameter	Estimate	Std. Error	t value	Pr($>|t|$)
a	118.1306	0.8709	135.649	<2e-16
b	70.5690	15.4602	4.565	$5.7e-06$

this relationship. A significant regression equation described in Table II was found $(F(1, 989) = 1.84x10^4, p < 2.2x10^{16})$, with an R^2 of 0.9535.

Considering time for finishing the task and number of collision avoidances, the descriptive statistics for the experiments with real UAVs is shown in Tables III and IV, and the corresponding boxplots are presented in Fig.8 and Fig. 9. Note that the behavior is similar in both testing environments.

In terms of *scalability*, the system completed the task for every experiment with nothing more than adding or removing agents without any modification on the configuration or the implementation. Also, considering *stability*, the system continued and finished the task even though one or more agents were force to stop, or because the battery charge was not enough to finish the job, and forced to return to the base, which happened a few times in the real setup.

It is important to emphasize that, despite the symmetric configuration for the field in both settings and the same starting points for the swarm of drones, sometimes they selected a different hole for seeding in an iteration as the nearest one,

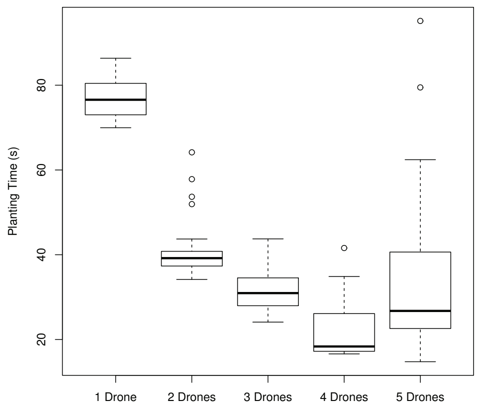

Fig. 8: Boxplots for each group of drones and planting time in real setting

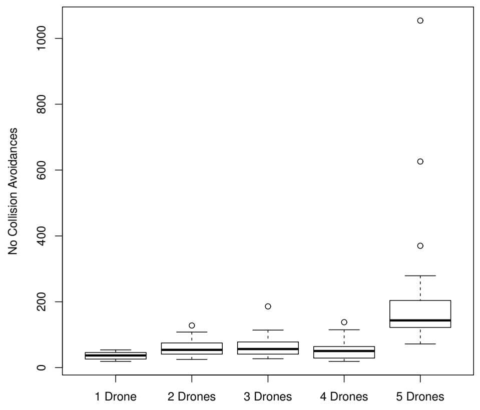

Fig. 9: Boxplots for each group of drones and number of collision avoidances in real setting

TABLE III: Descriptive statistic for each group of drones in real setting

No. Drones	Min.	Median	Mean	Max.	Std. Dev.
1	69.97	76.57	76.95	86.36	4.847
2	34.18	39.19	41.08	64.16	6.897
3	24.10	30.95	31.57	43.75	5.196
4	16.61	18.35	22.32	41.59	6.623
5	14.76	26.74	33.93	95.17	18.356

TABLE IV: Descriptive statistic for each group of drones for collision avoidances in real setting

No. Drones	Min.	Median	Mean	Max.	Std. Dev.
1	19.00	37.00	35.67	54.00	10.965
2	25.00	54.00	59.07	128.00	32.371
3	27.00	56.50	64.50	186.00	5.196
4	19.00	50.50	54.67	138.00	29.781
5	72.0	143.50	206.10	1054.00	192.372

because of the influence of possible collisions that forced them to change the target regarding a previous test. This behavior happened more frequently for real UAVs.

VI. Conclusions

This paper presents a centralized strategy for controlling a swarm of UAVs through an indirect local communication mechanism, which has shown low variability in performing a sowing seeding task in two scenarios, a simulation and in a real controlled environment. The results indicates that the time for performing the task decreases as the number of individuals increases in a non-linear way, which shows the behavior and effectiveness for an scalable system that improves or degrades in time when the number of agents is changed and possible collisions can occur. The system also shows stability when an agent is off from the swarm due to incidental events since the central control only updates the state for a place to be worked on the ground and processes the collision algorithm, the rest of the calculation resides on each agent and does not depend on each other to perform the task.

One of the consequences that comes from increasing the number of agents is the possible collisions among them and other obstacles. This work took into account a collision avoidance strategy, which has a considerable influence in the variability for the time to finish the task; also, it shows a linear behavior between the number of agents and collision avoidances with a variability that increases as well, even drastic outliers appeared in the results for the real environment when the agents were trying to repel each other at the same time with a large number of them, a case in which the robustness of the collision avoidance algorithm proves to work by not allowing impacts in such stressing situation in which hundreds of collision attempts were prevented, as presented in the results, due to the precise positioning provided by the centralized control and the strategy itself.

An important difference between the simulation and the real setting is the physical configuration of the agents. In the real setup, a drone tries to defeat its own inertia to change direction and get stabilized; also, the communication delay in this setup could affect the collision avoidance by reducing the radius r_c, which increases the possibilities of impacts because of the inertia. Such issues did not allow to follow the same pattern of movements on each iteration, despite of ground homogeneity.

For future work, the proposed strategy is intended to be implemented in a decentralized environment in which the approach resides on board at each agent. To accomplish this goal, the use of sensors and the independence from a motion capture system is necessary for real applications outside a controlled environment, also an implementation for the task of sowing a seed in the soil is necessary, together with suitable UAVs with greater payload.

Acknowledgement

This research was developed using the infrastructure provided by the ANDAMIOS Project supported by the Secretaría de Educación Superior, Ciencia, Tecnología e Innovación - SENESCYT of Ecuador.

References

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*, bonabeau1999, 1999.

[2] C. Anderson, “Agricultural Drones, Relatively cheap drones with advanced sensors and imaging capabilities are giving farmers new ways to increase yields and reduce crop damage,” *MIT Technology Review*, vol. 117, no. 3, 2014.

[3] C. Addison, “Drones for agriculture,” *ICT Update*, pp. 2–27, apr 2016.

[4] M. Mazur, “Six Ways Drones Are Revolutionizing Agriculture,” 2016. [Online]. Available: https://www.technologyreview.com/s/601935/six-ways-drones-are-revolutionizing-agriculture/

[5] F. G. Costa, J. Ueyama, T. Braun, G. Pessin, F. S. Osorio, and P. A. Vargas, “The use of unmanned aerial vehicles and wireless sensor network in agricultural applications,” *2012 IEEE International Geoscience and Remote Sensing Symposium*, pp. 5045–5048, 2012. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6352477

[6] Y. Meng, O. Kazeem, and J. C. Muller, “A Swarm Intelligence Based Coordination Algorithm for Distributed Multi-Agent Systems,” *Integration of Knowledge Intensive Multi-Agent Systems, 2007. KIMAS 2007. International Conference on*, pp. 294–299, 2007.

[7] D. Zhou and M. Schwager, “Assistive collision avoidance for quadrotor swarm teleoperation,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 1249–1254, 2016.

[8] D. H. Kim, H. Wang, and S. Shin, “Decentralized Control of Autonomous Swarm Systems Using Artificial Potential Functions: Analytical Design Guidelines,” *Journal of Intelligent and Robotic Systems*, vol. 45, no. 4, pp. 369–394, 2006. [Online]. Available: http://link.springer.com/10.1007/s10846-006-9050-8

[9] R. Sharma and D. Ghose, “Collision Avoidance Between UAV Clusters Using Swarm Intelligence,” *International Journal of Systems Science*, vol. 40, no. 5, pp. 521–538, 2009.

[10] I. Navarro, E. Di Mario, and A. Martino, “Distributed vs. Centralized Particle Swarm Optimization for Learning Flocking Behaviors,” *13th European Conference on Artificial Life (ECAL 2015)*, pp. 302–309, 2015. [Online]. Available: http://infoscience.epfl.ch/record/209289

[11] Y. Zhang, P. Agarwal, V. Bhatnagar, S. Balochian, and J. Yan, “Swarm Intelligence and Its Applications,” *Hindawi Publishing Corporation: The ScientificWorld Journal*, vol. 2013, p. 3, 2013. [Online]. Available: http://dx.doi.org/10.1155/2013/528069

[12] Y. Tan and Z.-y. Zheng, “Research Advance in Swarm Robotics,” *Defence Technology*, vol. 9, no. 1, pp. 18–39, 2013. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S221491471300024X

[13] Biocarbon Engineering, “Seed bomb drones: UK-based start-up to plant one billion trees in one year using drones - YouTube,” 2015. [Online]. Available: https://www.youtube.com/watch?v=r-4Zoo_qIcE

[14] National Geographic, “Drones and the Future of Farming — National Geographic - YouTube,” 2015. [Online]. Available: https://www.youtube.com/watch?v=v3YcZtlVrls

[15] Bitcraze, “Crazyflie 2.0,” 2016. [Online]. Available: https://www.bitcraze.io/crazyflie-2/

[16] M. W. Mueller, M. Hamer, and R. D’Andrea, “Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation,” in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, 2015, pp. 1730–1736.

[17] Unity, “Unity - Game Engine,” 2017. [Online]. Available: https://unity3d.com/