# Real-Time Flocking of Multiple-Quadrotor System of Systems

Osamah Saif[1] and Isabelle Fantoni[1] and Arturo Zavala-Río[2]

*Abstract*— **The subject of this paper is a real-time flocking control of multiple quadrotors in the context of system of systems. We believe that the most challenging aspect in multiple-quadrotor control is the interaction between quadrotors through sensing and preserving safe interdistances. The final objective is a collision-free flock of multiple quadrotors while navigating to a predefined destination. For this purpose, we develop control laws that are based on the consensus theory introduced by *Olfati-Saber* in [1]. Our control laws are designed in order to be compatible with experimental implementation and nonlinear model of quadrotors. Simulations and experiments using four quadrotors validate the performance of the proposed control laws. The convergence of interdistances between quadrotors to a desired value are maintained while navigating to a destination point.**

## I. Introduction

Flocking is a collective, harmonic and collision-free motion of animals. Biologists have studied flocks of birds, swarms of insects, herds of quadruped, and schools of fish, in order to understand the secret of this collective motion. Moreover, they have searched to discover motivations that lead animals to aggregate in groups [2] [3].

The amazing phenomenon of flocking has attracted control and robotics scientists, so they have tried to imitate it on robots platoons. One of the most advanced fields, that developed techniques similar to flocking, is flight formation control. Three of them are cited here: *Leader-follower*, *Virtual structure* and *Behavior-based control*. In the *leader-follower* structure, individuals in the formation follow one agent (or airplane) which is designated as a leader. A formation flight mission trajectory is loaded in the leader, and the followers track their leader. This structure is simple and widely implemented in multi-agent formation [4], [5]. Experimental work was conducted in [6] using this control structure. However, this control structure reveals some drawbacks. One of them is that the entire formation depends on one agent, so if there is a problem with the leader, the whole formation will be affected.

In the *virtual structure*, every agent in the formation has its own trajectory to follow. The overall trajectories form the desired formation. Trajectories are calculated in a central computer and sent to agents in the formation. Generally, no interactions between agents are considered. Examples of experimental works of such control strategy can be found in [7] and [8].

The two previous structures tend to be centralized methods of controlling multi-agent systems, which could have a weak performance when the number of agents in a formation increases. In fact, multi-agent systems control is considered as a system of systems issue. Technological advances in our ages motivate scientists to develop distributed methods that are capable to deal with system of systems challenges, such as operational and managerial independence, scalability, distributed coordination and synergism of control [9] [10] [11]. One of these methods is the *behavior-based control*, where each agent follows some rules to achieve the formation. The objective of the formation control is, therefore, broken down into small rules. In fact, this structure is inspired from the collective motion of animals. Among the first technical work on this structure is the distributed behavioral model by *Reynolds* [12]. Although Reynolds was specialized in computer graphics, his work inspired researchers in control theory and robotics to apply Reynolds rules in a theoretical and experimental framework. Reynolds inspired his rules from biologists study of animals collective motion. He considered that each individual in a flock should follow these rules in order to perform the flocking behavior. These rules are: 1) Collision Avoidance; 2) Velocity Matching, and 3) Flock Centering.

These rules were cited by several works in control theory and robotics such as [1], [13]. Authors used these rules as a base to develop theoretical framework and control strategies and algorithms for flocking of multi-agent dynamical systems. These theoretical results were not exploited in experimental works, to the best of our knowledge. This encourages us to extend the work in [1] to be applicable in real-time multiple quadrotors. In fact, we believe that distributed coordination, scalability and self-organization of the control laws in this work answer some challenges of system of systems engineering.

Quadrotors are a type of UAV devices that have received a great interest in the last decades. Maneuverability in 3-D space and low-cost experimentation of quadrotors render them an ideal platform for robotics research. Moreover, applications of quadrotors vary from surveillance and 3-D mapping of environments, to rescue and exploration.

A system of systems of multiple quadrotors could be employed in large-zone surveillance, rescue and search missions. Recently, researchers in robotics performed several experimental works on multiple-quadrotor control as in [7],

[1]O. Saif and I. Fantoni are with Sorbonne universités, Université de Technologie de Compiègne (UTC) – CNRS, UMR 7253 Heudiasyc, 60200 Compiègne, France. {osamah.saif/isabelle.fantoni}@hds.utc.fr

[2] A. Zavala-Río is with Instituto Potosino de Investigacion Cientifica y Tecnologica, Camino a la Presa San Jose 2055, Lomas 4a. Seccion 78216, San Luis Potosi , Mexico. azavala@ipicyt.edu.mx

This work was carried out and funded in the framework of the Labex MS2T. It was supported by the French Government, through the program "Investments for the future" managed by the National Agency for Research (Reference ANR-11-IDEX-0004-02)

[8] and [14]. The works in [7] and [8] consider multiple-quadrotor control as a problem of trajectory generation. Each quadrotor follows a collision-free trajectory, which is generated from a central base station. In the presence of disturbances, it is not clear that a collision-free formation could be ensured. Moreover, we believe that multiple-quadrotor control could be more challenging if it is implemented aboard quadrotors and if it considers interactions between them. On the other hand, the work in [14] did not deal with the nonlinear model of quadrotors. Instead, it uses a high level control that deals with an ideal double-integrator model, and the smooth output of this model is used to drive stabilized quadrotors. In our work, we try to deal directly with the nonlinear model of quadrotors, and then we apply the proposed control laws in the quadrotors to stabilize and navigate the flock.

Our interest in this work is to perform real-time flocking of multiple quadrotors in the context of system of systems. Our control method is based on the consensus theory introduced by *Olfati-Saber* in [1]. The flocking control law in [1] was mainly proposed for double-integrator linear system. In this work, we propose two modified versions of this control law aiming at being compatible with the nonlinear model of quadrotors and experimental works. The control law is run aboard each quadrotor in the flock. By running the control law, each quadrotor interacts with its neighbors to ensure a collision-free flocking.

The organization of this paper is: we address, firstly, the topology of multiple-quadrotor flocking and the dynamics of a quadrotor in section II. Then, the flocking algorithm of [1] and our two control laws are introduced in section III. Simulation and experimental results are presented in section IV. Finally, section V concludes with comments and proposed future works.

## II. PRELIMINARIES

### A. Topology of multi-quadrotor flocking

Graph theory is used to describe the topology of a multi-quadrotor system. A multi-quadrotor system is represented by an undirected graph $G = (\mathscr{V}, E)$, where $\mathscr{V}$ is a set of nodes $\mathscr{V} = \{1, 2, ..., M\}$, and $E$ is a set of edges $E \subseteq \{(i, j) : i, j \in \mathscr{V}, i \neq j\}$. Every node represents a quadrotor and edges depict the sensing between quadrotors. An adjacency matrix $A$ is an $M \times M$ matrix with elements $a_{ij} = 1$ if $(i, j) \in E$ and $a_{i,j} = 0$ otherwise. For more information about graph theory, the reader can refer to [15].

Before working on the dynamics of quadrotors, we need to represent our multi-quadrotor system in the Euclidean space. Therefore, to every node $i$ in the graph, a position vector $q_i \in \mathbb{R}^f$ is associated, where $f$ is the dimension of the space (example: $f = 2, 3$). The configuration of all nodes of the graph is defined by the vector $q = col(q_1, ..., q_n) \in \mathbb{R}^{fM}$.

A set of spacial neighbors of a quadrotor $i$ is defined by:

$$N_i = \left\{ j \in \mathscr{V} : \|q_j - q_i\| < c \right\} \tag{1}$$

where $\|.\|$ is the Euclidean norm, and $c$ is the interaction range. A position-induced graph $G(q) = (\mathscr{V}, E(q))$ is called a proximity net and is defined by $\mathscr{V}$ and the set of edges $E(q) = \left\{ (i, j) \in \mathscr{V} \times \mathscr{V} : \|q_j - q_i\| < c, j \neq i \right\}$.

The desired conformation of multiple quadrotors in a flock could be written as follows:

$$\|q_j - q_i\| = d \quad \forall j \in N_i(q) \tag{2}$$

where $d$ is the desired inter-distance. A proximity net that ensures the objective in (2) is defined as an "$\alpha$-*Lattice*". However, implicit inaccuracies give rise to an $\alpha$-*Lattice* with some edge-length uncertainty. This type of proximity net is called a "*quasi-$\alpha$-Lattice*" [1], and it is described by the following inequality:

$$-\delta < \|q_j - q_i\| - d < \delta \quad \forall (i, j) \in E(q) \tag{3}$$

where $\delta$ is the edge-length uncertainty.

### B. Quadrotor dynamics

In this section, we introduce nonlinear dynamics of a quadrotor. Several studies dealt with the modeling of quadrotors as in [16] and [17]. We model the quadrotor as a rigid body. Let $I = (I_x, I_y, I_z)$ be the global inertial frame, and let $B = (B_x, B_y, B_z)$ be the body-fixed frame. The nonlinear model of a quadrotor is given as follows:

$$\begin{array}{rcl} \dot{\xi} & = & \nu \\ m\ddot{\xi} & = & G + \mathbf{R}U \\ \dot{\eta} & = & W\Omega \\ J\dot{\Omega} & = & \Omega \times J\Omega + \tau \end{array} \tag{4}$$

where $\xi = [x, y, z]^T$ is the position of the center of mass of the quadrotor in the $I$ frame, $\nu = [\nu_x, \nu_y, \nu_z]$ is the vector of linear velocities in the $I$ frame, $\eta = [\phi, \theta, \psi]^T$ is the vector of Euler angles, $\Omega = [w_{B_x}, w_{B_y}, w_{B_z}]^T$ is the vector of the angular velocities in the $B$ frame, $m$ is the mass of the quadrotor, $G = [0, 0, -g]^T$ with $g$ being the gravitational acceleration, $U = [0, 0, F]^T$ is the thrust vector and $\tau = [\tau_\phi, \tau_\theta, \tau_\psi]^T$ is the torque vector.

$J$ is the moment of inertia diagonal matrix with $J_x$ $J_y$ and $J_z$ are diagonal components. $W$ is a transformation matrix between the angular velocities and the derivatives of Euler angles. $\mathbf{R}$ is the rotation matrix from the $B$ frame to the $I$ frame. The reader can refer to [17] for detailed expressions of $W$ and $\mathbf{R}$.

## III. FLOCKING ALGORITHMS OF MULTIPLE QUADROTORS

In this paper, we separate the control problem of multi-quadrotor in two parts. The first part is the control of internal dynamics of each quadrotor. We mean by internal dynamics, the altitude $z$ and the rotational dynamics of each quadrotor. This part will not be involved in the algorithm of flocking. The second part is the control of the $x, y$ translation and flocking dynamics of multiple quadrotors.

In fact, we specify a fixed desired altitude $z$ and heading $\psi$. Moreover, outputs of $x-y$ translation and flocking controllers are feedforwarded to the inputs of controllers of roll and pitch $\phi - \theta$ angles. Figure 1 shows the overall control architecture.

### A. Control of quadrotor internal dynamics

Before starting the description of the control strategy of a quadrotor in a flocking perspective, we begin by linearizing the nonlinear model (4) about the origin ($\xi = \mathbf{0}, \dot{\xi} = \mathbf{0}, \eta =$

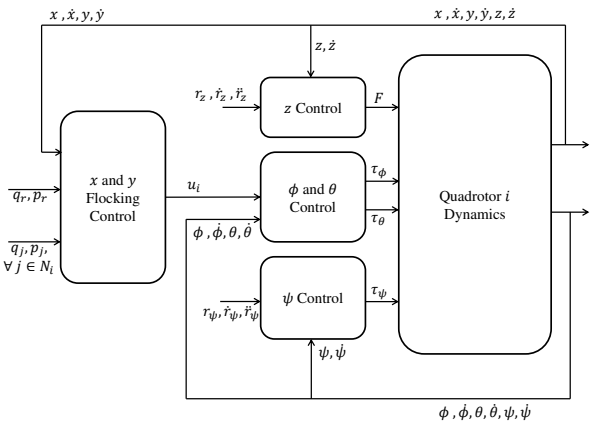


*Fig. 1: Control architecture of quadrotor in a flocking perspective. Internal dynamics are controlled separately from the $x-y$ translation and flocking dynamics*

$\mathbf{0}, \dot{\eta} = \mathbf{0}$). The result of linearization is given as follows:

$$\dot{x} = v_x \quad (5a) \qquad \dot{\phi} = w_{B_x} \quad (6a)$$
$$\dot{y} = v_y \quad (5b) \qquad \dot{\theta} = w_{B_y} \quad (6b)$$
$$\dot{z} = v_z \quad (5c) \qquad \dot{\psi} = w_{B_z} \quad (6c)$$
$$\dot{v}_x = (1/m)\theta \quad (5d) \qquad \dot{w}_{B_x} = (1/J_x)\tau_\phi \quad (6d)$$
$$\dot{v}_y = -(1/m)\phi \quad (5e) \qquad \dot{w}_{B_y} = (1/J_y)\tau_\theta \quad (6e)$$
$$\dot{v}_z = F \quad (5f) \qquad \dot{w}_{B_z} = (1/J_z)\tau_\psi \quad (6f)$$

Equations (5a) through (5f) represent the translational dynamics of the quadrotor and equations (6a)-(6f) represent the rotational dynamics.

The control of the quadrotor will be as follows. Firstly, we use PID controllers to control the $z$ dynamics (equations (5c) and (5f)) and the $\psi$ dynamics (equations (6c) and (6f)). The control inputs of these two subsystems are given by the general expression of a PID controller in a tracking perspective, as follows:

$$F = \ddot{r}_z + k_{pz}(r_z - z) + k_{dz}(\dot{r}_z - \dot{z}) + k_{iz}\int (r_z - z)\,\mathrm{d}t \quad (7)$$

$$\tau_\psi = \ddot{r}_\psi + k_{p\psi}(r_\psi - \psi) + k_{d\psi}(\dot{r}_\psi - \dot{\psi}) + k_{i\psi}\int (r_\psi - \psi)\,\mathrm{d}t \quad (8)$$

where $r_z$ and $r_\psi$ are the desired altitude and heading of the quadrotor, and the constants $k_{(.)} > 0$.

Secondly, the remaining equations represent the $x-y$ translational dynamics and the $\phi-\theta$ rotational dynamics. To control these dynamics we use an approach similar to the backstepping technique. This approach is widely used in the control of quadrotors [17]. First, we consider $\phi$ and $\theta$ in equations (5d) and (5e) as virtual control inputs of the translational dynamics. Then, we design a controller for the rotational dynamics (equations (6a)-(6f)), which has a double-integrator form. For this purpose, we use the nested saturation approach [18], [19], [20] and [21]. Hence, the control inputs that stabilize the $\phi-\theta$ rotational dynamics are given as follows:

$$\tau_\phi = -Sat_{\phi_1}\left(k_{\phi_1}\dot{\phi} + Sat_{\phi_2}\left(k_{\phi_2}\dot{\phi} + k_{\phi_1}k_{\phi_2}\left(\phi - r_\phi\right)\right)\right) \quad (9)$$

$$\tau_\theta = -Sat_{\theta_1}\left(k_{\theta_1}\dot{\theta} + Sat_{\theta_2}\left(k_{\theta_2}\dot{\theta} + k_{\theta_1}k_{\theta_2}\left(\theta - r_\theta\right)\right)\right) \quad (10)$$

where $Sat_{\alpha_i}(x) = sign(x)min(|x|, \alpha_i)$ with $(i = 1,2)$, $\alpha_i = \phi_i$ or $\theta_i$, with $\alpha_i$ being a real positive constant. The constants $k_{(.)}$ are tuning gains. $r_\phi$ and $r_\theta$ are desired references, which are the outputs of the $x-y$ position controllers.

### *B. Flocking and translation control of multiple quadrotors*

The design of the $x-y$ controllers will be similar to the flocking algorithm in [1]. In equations ((5a), (5b), (5d), (5e)), we take $q_i = [x\ y]^T$, $p_i = [v_x\ v_y]^T$, and $u_i = [\frac{1}{m}\theta\ \frac{-1}{m}\phi]^T$. Therefore, the translational dynamics could be written as follows:

$$\begin{aligned}\dot{q}_i &= p_i \\ \dot{p}_i &= u_i\end{aligned} \quad (11)$$

In this section, we discuss the control of such double-integrator translation system from a flocking perspective. We begin by introducing the basic principles of controlling a multi-agent system. The controlling algorithm is introduced by Olfati-Saber in [1]. A "$\sigma$-norm" is a map $\mathbb{R}^n \to \mathbb{R}^+$ of a vector $z \in \mathbb{R}^n$, defined as:

$$\|z\|_\sigma = \frac{1}{\varepsilon}\left[\sqrt{1+\varepsilon\|z\|^2} - 1\right] \quad (12)$$

where $\varepsilon > 0$ and $\mathbb{R}^+$ is the set of non-negative real numbers. The gradient of $\sigma$-norm is defined by:

$$\sigma_\varepsilon(z) \triangleq \nabla_z\|z\|_\sigma = \frac{z}{\sqrt{1+\varepsilon\|z\|^2}} = \frac{z}{1+\varepsilon\|z\|_\sigma} \quad (13)$$

In fact, $\sigma$-norm is not a norm but its importance is that it is differentiable everywhere, unlike the Euclidean norm that is not differentiable at $z = 0$.

A spacial adjacency matrix is defined as $A(q) = [a_{ij}(q)]$ where:

$$a_{ij}(q) = \begin{cases} 0 & \text{if } i = j \\ \rho_h(\|q_j - q_i\|_\sigma / \|c\|_\sigma) & \text{if } j \neq i \end{cases} \quad (14)$$

The bump function $\rho_h : \mathbb{R}^+ \to [0,1]$ with $h \in (0,1)$ is defined as:

$$\rho_h(z) = \begin{cases} 1 & \text{if } z \in [0,h) \\ \frac{1}{2}\left[1 + \cos\left(\pi\frac{z-h}{1-h}\right)\right] & \text{if } z \in [h,1] \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

A smooth collective potential function is used to design the flocking algorithm of multiple quadrotors. This function is given as follows:

$$V(q) = \frac{1}{2}\sum_i \sum_{j\neq i} \Psi_\alpha(\|q_j - q_i\|_\sigma) \quad (16)$$

where

$$\Psi_\alpha(z) = \int_{d_\alpha}^{z} \Phi_\alpha(s)ds \quad (17)$$

$\Phi_\alpha$ is defined by:

$$\begin{aligned}\Phi_\alpha(z) &= \rho_h(z/c_\alpha)\Phi(z - d_\alpha) \\ \Phi(z) &= \tfrac{1}{2}[(a+b)\sigma_1(z+e) + (a-b)]\end{aligned} \quad (18)$$

with $\sigma_1(z) = z/\sqrt{1+z^2}$. The function $\Phi(z)$ is uneven and sigmoidal, with $0 < a \leq b$ and $e = |a-b|/\sqrt{4ab}$ that ensures $\Phi(0) = 0$.

It follows from the above formulas that $\Psi_\alpha(z)$ is a smooth pairwise repulsive/attractive potential function. It has

a minimum at $z = d_\alpha = \|d\|_\sigma$, and it has a finite cut-off at $c_\alpha = \|c\|_\sigma$. The finite cut-off feature of this function is a fundamental source of scalability of the flocking algorithm [1]. Moreover, every local minimum of $V(q)$ is an $\alpha$-lattice.

The control law introduced in [1], which is applied on each agent with linear dynamics, and that ensures an *$\alpha$-lattice* flock and navigation, is given as follows:

$$\begin{aligned} u_i &= \sum_{j\in N_i} [\Phi_\alpha(\|q_j - q_i\|_\sigma)\mathbf{n}_{ij} + a_{ij}(q)(p_j - p_i)] \\ &+ f_i^\gamma(q_i, p_i, q_r, p_r) \end{aligned} \tag{19}$$

where $f_i^\gamma(q_i, p_i, q_r, p_r) = -c_1(q_i - q_r) - c_2(p_i - p_r)$, $c_1, c_2 > 0$, and $\mathbf{n}_{ij} = \sigma_\varepsilon(q_j - q_i)$ as in equation (13).

This control law is composed of three terms. The first is the gradient-based term, which ensures the interdistance regulation between agents. The second is the velocity consensus term, which is analog to a derivative controller in a conventional PD control law. The last term $f_i^\gamma(.)$ is the navigational or the translational feedback control, with $q_r$ and $p_r$ being the desired position and velocity to be tracked. Moreover, the first and the second terms ensure the aggregation of every agent with its neighbors and the conservation of a collision-free flocking. The navigational feedback leads the whole flock to track a predefined objective trajectory or destination point. The objective trajectory is known by every agent in the flock, which ensures a fragmentation-free flocking.

We have applied such controller in simulation (section IV), and we have observed that this control law could not be applied directly on nonlinear systems, such as quadrotors. In fact, the simulation results of this control law on the nonlinear dynamics of quadrotors show oscillating movement of distances between quadrotors. This could be explained by the fact that the control law in (19) was applied on double-integrator linear models without considering uncertainties. We believe that, the elegant control law in (19) could be applied on nonlinear systems if we add some tuning gains to its gradient-based and velocity consensus terms. The additional gains will compensate for uncertainties of nonlinear model. The modified control law is given in the following equation:

$$\begin{aligned} u_i = \sum_{j\in N_i} \Big[ & K_p \Phi_\alpha(\|q_j - q_i\|_\sigma)\mathbf{n}_{ij} \\ & + K_d a_{ij}(q)(p_j - p_i) \Big] + f_i^\gamma(q_i, p_i, q_r, p_r) \end{aligned} \tag{20}$$

where $K_p, K_d > 0$ are tuning gains and their values depend on the quadrotor device. $K_p$ and $K_d$ are user defined. They give a relative freedom to the user to apply the control law on different quadrotor devices.

### C. Robust flocking control

In the previous section, we proposed a modified version of Olfati-Saber flocking control law. This control law was tested in a real-time experimental setup and showed good results. However, real-time applications always suffer of perturbations, and need more robust control laws. Perturbations are caused generally by the wind flown from the rotors of quadrotors. In real-time experiments we noticed steady-state errors in the distances between quadrotors. Therefore, in this section, we present an alternative version of (20), intended to eliminate the steady-state errors.

In control theory, one of the ways used to eliminate steady-state errors is to add an integral action to the control law [22]. Therefore, our complemented control law is written as follows:

$$\begin{aligned} u_i = \sum_{j\in N_i} \Big[ & K_p \Phi_\alpha(\|q_j - q_i\|_\sigma)\mathbf{n}_{ij} + K_d a_{ij}(q)(p_j - p_i) \\ & + K_i \int \Phi_\alpha(\|q_j - q_i\|_\sigma)\, \mathbf{n}_{ij}\, \mathrm{d}t \Big] + f_i^\gamma(q_i, p_i, q_r, p_r) \end{aligned} \tag{21}$$

where $K_i$ is the integral-action gain, which is tuned by the user. This alternative control law shows good results in real-time experiments, which complements *Olfati-Saber* approach. The steady-state errors are efficiently reduced.

## IV. Simulation and Experiments

In this section, we present our simulation results and real-time experiments of multi-quadrotors flocking. By using a simulator of multiple quadrotors, we apply the control law in equation (19). Real-time experiments show the results of the application of our complemented control laws (20) and (21). Moreover, we compare our results with another work of the authors in [23]. Videos of the simulation and real-time experiments are available in the supplementary materials or at the link in [24].

### A. Simulation

The simulation results show the application of Olfati-Saber control law (19). Generally, before performing risky real-time experiments, and especially with multi-quadrotors, we perform numerical implementations on a simulator.

Heudiasyc laboratory has developed a PC-based simulator of flock of multiple quadrotors. *C++* codes written in the simulator and in the embedded system of each quadrotor are the same. For this purpose, the PC is run under Linux as in the quadrotors. In the simulator, virtual sensors and actuators are connected to the discrete nonlinear model of quadrotor in (4). As a result, all quadrotors' states are calculated at each instant of time. Moreover, quadrotors evolve in a 3-D virtual environment, thanks to Irrlicht engine. The program in the simulator is connected to a base-station control program. The base station records and draws measurements. Moreover, it is used to start and end simulation and to set the parameters of quadrotors and control laws. Figure 2 shows the simulation result of applying the control law in (19) on four quadrotors. This figure represents the distances between quadrotors over the time, and the legend $\mathrm{d}_{ij}$ means the distance between quadrotor $i$ and $j$. The intended scenario is as follows: quadrotors start in their initial positions and move toward a predefined $q_r$ position. For the numerical application, we set $a = b = 5$, the desired distance between neighbors is $d = 1$, $c = 1.2d$, $\varepsilon = 0.1$ and $h = 0.2$. In this figure, the distance between quadrotors 1 and 3 is not converging to the desired value because this distance exceeds the interaction range $c = 1.2d$. Moreover, We note the oscillated behavior of the quadrotors during flight. This is caused by the absence of

gains in the gradient-based and the velocity consensus terms in (19). Since this results are risky, we avoided experimental works with this control law.

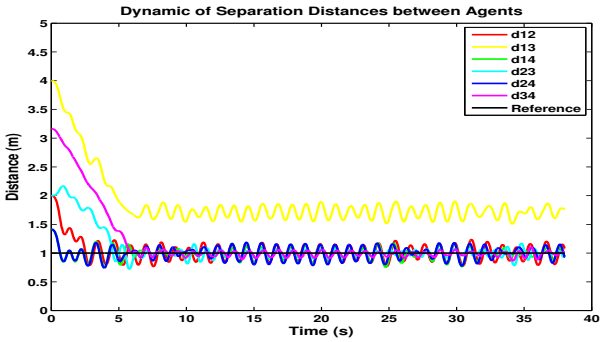


*Fig. 2: Interdistances between 4 quadrotors in the simulator by using Olfati-Saber algorithm (19). Once the control law of flocking is launched in each quadrotors, the interactions start and the distances converge with oscillation*

### B. Experimental platform

In our experiments, our platform is the quadrotor ArDrone2 from Parrot [25]. Using an SDK provided by Parrot, this platform is designed to be controlled, either from a smartphone, a PC through WiFi, or directly by running a program on the quadrotor via socket. The utilization of SDK prevents us to use our control laws to stabilize the quadrotor, since the drone has its own control laws designed by Parrot.

We solved this problem through the work of teams from TU Delft university on ARDrone 2 and their autopilot *Paparazzi* [26]. They managed to decode the communication protocols between the main processor of the quadrotor and its sensors and motors. With these protocols, it is possible to directly control the quadrotor and read the raw data from each sensor. By incorporating these protocols into our own software framework for quadrotors, we managed to replace the programs of manufacturer by our own control laws.

The ARDrone 2 is thus mainly used for its material part, whose characteristics are: 1GHz 32 bit ARM Cortex A8 processor, 128 MB RAM, 128MB Flash, WiFi, 3 axis accelerometer, 3 axis gyroscope, 3 axis magnetometer, Pressure sensor, Ultrasound sensors (altitude $< 6m$) and 4 brushless motors. The inertial sensors are used in a complementary filter [27] to estimate the orientation of the quadrotor.

Experiments are performed in an indoor environment using Optitrack motion capture system [28]. The system senses the pose of quadrotors at 100 Hz. This information is sent to the base-station PC and then forwarded through Wifi to one of quadrotors. This quadrotor broadcasts the pose information to all quadrotors. Every quadrotor, then, knows the pose of all quadrotors in the flock and can estimate also velocities by differentiating the poses. In all experiments, we use the Optitrack frame of reference as our global frame $I$.

We emphasize here that our control laws are run aboard on the quadrotors. Moreover, our control laws need only the relative distances to the neighboring quadrotors. Since we do not have sensors that measure the relative distances to neighbors, we use the Optitrack system as an alternative. Thus, we calculate the relative distances, aboard on, by using received positions. Moreover, $a_{ij}(.)$ function in (14) is used to limit the interaction range between quadrotors.

### C. Experimental results

In this section, we present real-time experiments of multiple-quadrotor flocking. We show the results of two experiments using the control laws in (20) and (21). Moreover, we compare these results to a previous work of the authors in [23].

In all of the following experiments, quadrotors takeoff from their initial positions to the same defined altitude $r_z = 1m$. The desired yaw angles are set to $r_\psi = 0$ for the whole experiment duration. The formation control law is then launched to form the desired conformation. Finally, the quadrotors land after sufficient time.

In the first experiment, we use four quadrotors to form a *quasi-$\alpha$-lattice*. We apply our first improved control law in (20). The destination point is defined as the origin of the frame $I$, $q_r = [0\ 0]^T$. In this experiment we set $K_p = 0.25, K_d = 0.3, c_1 = 0.1, c_2 = 0.2$, $\varepsilon = 0.1$, $h = 0.2$, $c = 2$, the desired distance between neighbors is $d = 1.5$ and the parameters $a = b = 1$.

Figure 3 (**left**) shows the result of using the first complemented control law in (20) in the first real-time experiment. The figure exhibits the distances between quadrotors over the time. In the experiment, the performance of this control law is improved compared to the one in (19). We note, however, a steady-state error in the interdistances between quadrotors, i.e. the desired interdistances are not completely reached. This steady-state error could be explained by the presence of continuous perturbations in the real-time experiment. One of the sources of these perturbations is the downwash of rotor blades.

Figure 3 (**right**) shows the trajectories of quadrotors of this experiment. Quadrotors start at their initial positions designated as black diamonds. Then, they start moving toward the desired destination while they avoid collision with their neighbors. A *quasi-$\alpha$-lattice* is finally formed. In

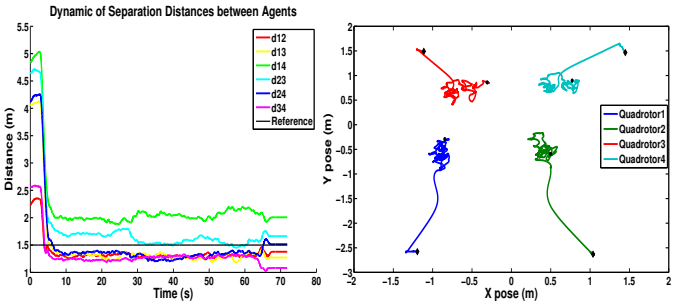


*Fig. 3: (**Left**) Interdistances between 4 quadrotors in the real-time experiment by using the control law in (20). Once the control law of flocking is launched in each quadrotors, the interactions start and the distances converge. Finally, the quadrotors land after t = 60s. (**Right**) Trajectories of 4 quadrotors in the real-time experiment by using the control law in (20). Quadrotors move from their initial positions to the destination point, gather around the center and form a* quasi-$\alpha$-lattice

the second experiment, we apply our control law in (21). The destination point is designated to be $q_r = [1\ 1]^T$. The parameters of this control law is set as $K_p = 0.25, K_d =$

0.3, $K_i = 0.09$, $c_1 = 0.1$, $c_2 = 0.2$, $\varepsilon = 0.1$, $h = 0.2$, $c = 2$, the desired distance between neighbors is $d = 1.5$ and the parameters $a = b = 1$.

Figure 4 (**left**) shows the result of the second experiment using the control law (21). The quadrotors go toward the destination point, and the distances between neighbors converge to the desired value. In this experiment, steady-state errors are eliminated, thanks to the integral action in the alternative control law (21). The distance between quadrotors 1 and 4 is greater than $d = 1.5$ because this distance exceeds the interaction range $c = 2m$. Figure 4 (**right**) shows the trajectories of the quadrotors navigating to the destination point during the experiment.

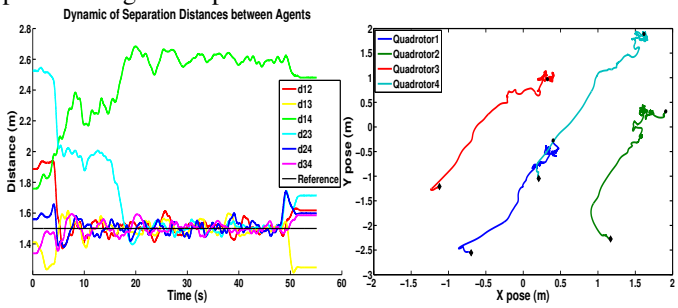


*Fig. 4: (**Left**) Interdistances between 4 quadrotors in the real-time experiment by using the control law in (21). After launching the flocking control law, interactions start between quadrotors and distances converge. Finally, the quadrotors land after t = 47s. (**Right**) Trajectories of 4 quadrotors, navigating to the destination point. The quadrotors start at the left down side of the figure and tend to the right top side. A* quasi-$\alpha$-lattice *is formed*

## V. Conclusion

In this paper, we showed real-time experiments of multiple-quadrotor flocking in the context of system of systems. Experimental results showed better performance of the proposed control laws compared to simulation results from previous schemes. Moreover, our results showed an improved performance compared to previous work of authors in [23]. In addition, our control laws were run aboard on quadrotors and each quadrotor interacted with its neighbors in the flock. Our work was based on the approach of *Olfati-Saber* in [1]. We proposed alternative control laws in order to be compatible with the nonlinear model and real-time experiments of multiple quadrotors.

Our future work will focus on the improvement of our experimental platform for outdoor experiments. In addition, we will work on the navigation of multiple quadrotors with obstacle avoidance.

## Acknowledgment

The authors would like to thank Guillaume Sanahuja, who is research engineer in Heudiasyc laboratory, for his efforts in developing the Simulator and the experimental platform and for his help in implementing the control laws in the quadrotors.

## References

[1] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, pp. 401 – 420, march 2006.

[2] B. Partridge, "The structure and function of fish schools," *Scientific American*, vol. 246, no. 06, pp. 114–123, 1982.

[3] I. D. Couzin, "Collective cognition in animal groups," *Trends in Cognitive Sciences*, vol. 13, no. 1, pp. 36 – 43, 2009.

[4] J.-A. Guerrero and R. Lozano, *UAV flight formation control*. John Wiley-ISTE, 2012.

[5] M. Chiaramonti, F. Giulietti, and G. Mengali, "Formation control laws for autonomous flight vehicles," in *14th Mediterranean Conference on Control and Automation. MED '06*, pp. 1 –5, june 2006.

[6] G. Vasarhelyi, C. Viragh, G. Somorjai, N. Tarcai, T. Szorenyi, T. Nepusz, and T. Vicsek, "Outdoor flocking and formation flight with autonomous aerial robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014), 2014*, pp. 3866–3873, Sept 2014.

[7] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors.," *Autonomous Robots*, vol. 35, pp. 287–300, 2013.

[8] A. Schollig, F. Augugliaro, S. Lupashin, and R. D'Andrea, "Synchronizing the motion of a quadrocopter to music," in *IEEE International Conference on Robotics and Automation (ICRA), 2010, Anchorage, Alaska*, pp. 3355–3360, May 2010.

[9] M. Jamshidi, ed., *Systems of Systems Engineering: Principles and Applications*. CRC Press, November 2008.

[10] M. Jamshidi, "System of systems - innovations for 21st century," in *IEEE Region 10 and the Third international Conference on Industrial and Information Systems. ICIIS*, pp. 6–7, Dec 2008.

[11] P. Gazi, M. Jamshidi, A. Jevtic, and D. Andina, "A mechatronic system design case study: Control of a robotic swarm using networked control algorithms," in *4th Annual IEEE Systems Conference*, pp. 169–173, April 2010.

[12] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," in *Computer Graphics*, pp. 25–34, 1987.

[13] H. Tanner, A. Jadbabaie, and G. Pappas, "Flocking in fixed and switching networks," *IEEE Transactions on Automatic Control*, vol. 52, pp. 863–868, May 2007.

[14] A. Franchi, C. Secchi, M. Ryll, H. Bulthoff, and P. Giordano, "Shared control : Balancing autonomy and human assistance with a group of quadrotor uavs," *IEEE Robotics Automation Magazine*, vol. 19, pp. 57–68, Sept 2012.

[15] R. Diestel, *Graph Theory*, vol. 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, third ed., 2005.

[16] R. Lozano, *Unmanned Aerial Vehicles Embedded Control*. John Wiley-ISTE Ltd, 2010.

[17] S. Bouabdallah and R. Siegwart, "Full control of a quadrotor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, California, USA. IROS 2007*, pp. 153–158, Oct 2007.

[18] A. R. Teel, "Global stabilization and restricted tracking for multiple integrator with bounded control," *Systems and Control Letters*, vol. 18, pp. 165–171, 1992.

[19] E. Johnson and S. Kannan, "Nested saturation with guaranteed real poles," in *Proceedings of the 2003 American Control Conference, Colorado, USA*, vol. 1, pp. 497–502 vol.1, June 2003.

[20] G. Sanahuja, *Commande et localisation embarquée d'un drone aérien en utilisant la vision*. PhD thesis, Université de technologie de compiègne, 2010.

[21] F. Kendoul, D. Lara, I. Fantoni, and R. Lozano, "Real-time nonlinear embedded control for an autonomous quadrotor helicopter," *Journal of Guidance, Control, and Dynamics*, vol. 30, pp. 1049–1061, 2007.

[22] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, Jan. 2002.

[23] O. Saif, I. Fantoni, and A. Zavala-Rio, "Flocking of multiple unmanned aerial vehicles by lqr control," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, USA*, pp. 222–228, May 2014.

[24] `https://www.hds.utc.fr/~ifantoni/dokuwiki/doku.php?id=en:videos`.

[25] `http://ardrone2.parrot.com/`.

[26] `http://wiki.paparazziuav.org/wiki/AR_Drone_2/getting_started`.

[27] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Transactions on Automatic Control*, vol. 53, pp. 1203–1218, June 2008.

[28] "Optitrack motion capture systems." `https://www.naturalpoint.com/optitrack/`.