Flight Controller Optimization of Unmanned Aerial Vehicles using a Particle Swarm Algorithm

Nicolas Gomez[1], Victor Gomez[1], Enrique Paiva[1], Jorge Rodas[1] and Raul Gregor[1]

***Abstract*— In this paper, a simultaneous calibration algorithm of the parameters of the attitude and altitude control for an unmanned aerial vehicle (UAV) is proposed. The algorithm is based on the multi-objective particle swarm optimization (MOPSO) technique. This algorithm is implemented by using the free PX4 software for the Pixhawk2 controller. The behavior of the UAV is simulated given its physical characteristics by means of a non-linear model and a search of the controller parameters. This latter is based on a proportional (P) position controller in cascade with a proportional-integral-derivative (PID) speed controller of its height and each one of its Euler angles. To perform this search, the PID gains K_{p1}, K_{p2}, K_i and K_d of each of the degrees of freedom are used to define vectors considered particle positions by the MOPSO algorithm, which moves them through a search space to find sets of optimum values according to Pareto, or the Pareto Front. The search is carried out based exclusively on Pareto dominance concepts, comparing parameters of step responses (overshoot, rise time, root-mean-square error) of each of the degrees of freedom. In order to show the efficiency of the proposal, simulation results are provided by using the calibration methodology obtaining good results.**

I. Introduction

Unmanned aerial vehicles (UAVs) usually have on-board computers that are used to stabilize and guide their flight. This latter is controlled by radio frequency by using a manual controller, a pre-defined trajectory or an algorithm to accomplish specific tasks. The range of possible applications presented by the UAVs is broad, for the ease they offer when it comes to reaching places of difficult or dangerous access for human beings, due to the height or hostile conditions of the environment [1]. Moreover, UAVs have attracted as well the control and automation community, which has to lead to new controller development [2]– [5].

Today, there are a huge number of companies that develop UAVs. This is one of the main reason why its cost decrease progressively. In addition, there are several communities dedicated to the collaborative development of free software and hardware for personal, commercial, and research applications of the UAVs. One of the control devices that arise from these communities is the well-known Pixhawk2, which uses the free PX4 autopilot software. This device is a low-cost controller that has great support from the open-source community and it is applicable to different types of UAVs. One issue that has to be taken into account to use the PX4-based controller is that it must be calibrated first. The PX4 has a proportional-integral-derivative (PID) controller for its attitude and altitude control that comes as default in its firmware. This controller is commonly calibrated with an automatic calibration algorithm (autotune) for proper operation, with the necessity of a fine-tuning performed empirically [6]– [8].

Automatic calibration requires a flight test, with a prepared custom field needed for this end. Such calibration is normally performed outside, resulting in the risk of the UAV destabilizing in flight and ending up crashing due to wind or other perturbations. In addition to all this, since a subsequent manual calibration is necessary, the entire task can take a long time, arising the possibility of depleting the UAV battery charge only in its calibration.

In order to offer an alternative to the above-mentioned issue, this paper proposes and implements a PID calibration algorithm based on the multiple objective particle swarm optimization (MOPSO) for Px4-based UAVs. A similar study has been proposed in [9] to obtain the parameters of the attitude controller of a UAV using a linearized simulation of the UAV plant. The optimization was carried out by reducing a single objective function composed of a weighted sum of the objective values in each axis of the UAV's attitude. The objectives considered in each axis were the overshoot, the steady-state error, and the difference of settlement time with rising time. However, in this work, a nonlinear model will be used for plant simulation. In addition, the proposed optimization method will not consider a weighted sum of the objective values, since this has as a disadvantage the prioritization of some of the objective values only, making it difficult to obtain the best values for all of the considered objectives.

This paper takes as a starting point [10], [11] to theoretically perform the tuning PID procedure for a quadrotor UAV. Obtained simulation results are based on a commercial FlameWheel F450 brand quadcopter.

II. Mathematical model

A. Descriptive equations of the model

The system of interest in this paper is the quadrotor. The model that describes the system is characterized for being nonlinear. The system has, in total, 12 states that are described next. Three position states ("north" p_n, "east" p_e, "down" p_d), aligned with inertial frame of reference. In addition, three linear velocity states (u, v, w) and three angular velocity states (p, q, r) aligned with the body frame. At last, three angular position states ("roll" ϕ, "pitch" θ,

[1]Nicolas Gomez, Victor Gomez, Enrique Paiva, Jorge Rodas and Raul Gregor are with the Laboratory of Power and Control Systems, Facultad de Ingeniería, Universidad Nacional de Asunción, Luque, Paraguay. ni_co182@hotmail.com, sebasg7@gmail.com, engapaga222@hotmail.com, jrodas@ing.una.py, rgregor@ing.una.py

"yaw" ψ), aligned with the vehicle reference frame. The equations that represent a generic UAV [12] can be described as follows:

$$\begin{bmatrix}\dot{p_n}\\ \dot{p_e}\\ \dot{p_d}\end{bmatrix} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (1)$$

$$\begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix} = \begin{bmatrix} r\,v - q\,w \\ p\,w - r\,u \\ q\,u - p\,v \end{bmatrix} + \mathbf{g}\begin{bmatrix} -s_\theta \\ c_\theta\, s_\phi \\ c_\theta\, c_\phi \end{bmatrix} - \frac{1}{\mathbf{m}}\begin{bmatrix}0\\0\\f\end{bmatrix} \quad (2)$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix} 1 & s_\phi\, s_\theta/c_\theta & -c_\phi\, s_\theta/c_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi/c_\theta & c_\phi/c_\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (3)$$

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \mathbf{I}^{-1}\left(\begin{bmatrix}p\\q\\r\end{bmatrix} \times \mathbf{I}\begin{bmatrix}p\\q\\r\end{bmatrix} + \begin{bmatrix}\tau_\phi\\ \tau_\theta\\ \tau_\psi\end{bmatrix}\right) \quad (4)$$

where $c_x = \cos(x)$ and $s_x = \sin(x)$. $\mathbf{m}$, $\mathbf{g}$ and $\mathbf{I}$ denote the total mass of the UAV, the acceleration of gravity and the inertia matrix, respectively. $\tau = [\tau_\phi, \tau_\theta, \tau_\psi]^T$ denotes the total external moments applied and is obtained from (7). By using the UAV's geometry represented in Fig. 1, the total thrust f can be computed as $f_1 + f_2 + f_3 + f_4$. The torque values τ_1 and τ_2 are computed according to the axes' directions in Fig. 2. This corresponds to (5)-(7), where d denotes the UAV diameter and m_i denotes the torque generated by each propeller. Aerodynamic forces are considered perturbations of the system, since they have small influence compared to the ones produced by the motors and the mass of the UAV.

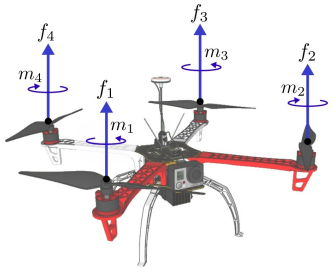

Fig. 1. Position of each rotor of the quadrotor.

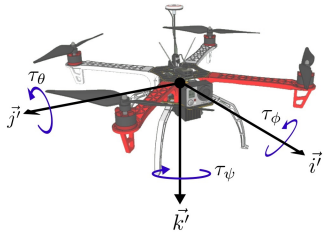

Fig. 2. Quadrotor orientation.

$$\begin{bmatrix}\tau_{\phi 1}\\ \tau_{\theta 1}\\ \tau_{\psi 1}\end{bmatrix} = \frac{d}{2}\begin{bmatrix} \frac{-\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \quad (5)$$

$$\begin{bmatrix}\tau_{\phi 2}\\ \tau_{\theta 2}\\ \tau_{\psi 2}\end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -m_1 + m_2 - m_3 + m_4 \end{bmatrix} \quad (6)$$

$$\begin{bmatrix}\tau_\phi\\ \tau_\theta\\ \tau_\psi\end{bmatrix} = \begin{bmatrix}\tau_{\phi 1}\\ \tau_{\theta 1}\\ \tau_{\psi 1}\end{bmatrix} + \begin{bmatrix}\tau_{\phi 2}\\ \tau_{\theta 2}\\ \tau_{\psi 2}\end{bmatrix} \quad (7)$$

B. Determination of UAV body parameters

UAV's mass was found by using a scale and is $\mathbf{m} = 1.3240$ kg. The diameter d of the UAV is 46.7 cm . Since the UAV is highly symmetrical, it is assumed that the moment of inertia tensor $\mathbf{I}$ is diagonal. The moment of inertia of a UAV from its center of gravity with respect to its axes of rotation can be measured with the bifilar pendulum method, described in [13]. To identify the main diagonal of the UAV inertia tensor, at least three tests must be done along the axes of I_x (roll), I_y (pitch) and I_z (yaw). The results so obtained were $I_x = 0.0124$ kg m^2, $I_y = 0.0130$ kg m^2 and $I_z = 0.0237$ kg m^2.

C. Determination of UAV propulsive parameters

Due to the low speed of the incident wind in each motor in a multicopter, and the high speed of response of the motors, it was decided to perform a relatively simple function to determine the thrust and torque produced by each motor based on the value of the duty cycle X_i of the pulse-width-modulated signals of each motor. These functions, shown in (8) and (9), have a polynomial form. Where f_i and m_i are the n^{th} grade polynomial functions of the thrust in N and torque in N m respectively with respect to the $X_i \in [1000, 2000]$. In this work, $n = 5$ is considered.

$$f_i = \sum_{k=0}^{k=n} X_i^k P_t^k \quad (8)$$

$$m_i = \sum_{k=0}^{k=n} X_i^k P_m^k \quad (9)$$

These functions are obtained through a series of tests of each motor in which several points are obtained, after which a polynomial approximation is made to obtain the coefficients P_t^k and P_m^k.

Four thrust and torque tests have been carried out on each of all four motor-propeller pairs of the UAV. In each test, 20 operating points were extracted. After that, an average of the values obtained at each working point of the tests carried out in each motor-propeller pair have been made. After obtaining these averages, a polynomial regression was made with the order n specified above, obtaining the coefficients of the polynomials that define the functions f_i and m_i respectively.

The polynomials obtained for the thrust and torque with this method are:

$$P_t = [-2.315^{-14}, 1.680^{-10}, -4.860^{-7}, 0.001, -0.505, 143],$$

$$P_m = [-3.32^{-16}, 2.41^{-12}, -7.01^{-9}, 1.02^{-5}, -0.007, 2.07].$$

D. Studied controller

Fig. 3 shows the PX4 attitude control structure [6]. It consists of a control block of the Euler angles of P-type in closed-loop, in cascade with an angular velocity control of the PID-type in closed-loop.

The altitude control is analogous to the attitude one, although a constant "hovering thrust control" value is added to the output of this controller. That thrust control value is such that the UAV could fly without acceleration on the altitude axis when this controller has a zero output. For this work, the hovering thrust control value is held with its default value. All controllers operate at the same time in parallel. The input reference are the desired Euler's angles in degrees and the altitude in meters, as well as the rate of change of every one of them in degrees per second and meters per second, respectively. The control efforts are the moments in each direction that, after being projected to the body frame, are used to find the thrust of each motor-propeller pair. The gain values K_{p1}, K_{p2}, K_i and K_d are the control parameters to be optimized for each controller in this paper by using MOPSO algorithm. As there are 4 degrees of freedom (roll, pitch, yaw, and vertical thrust), there are a total of 16 parameters.

E. Quadrotor UAV modelling test

For comparison of the modeled and real-life UAV behaviors, a series of flights of the UAV where made to manually find proper roll axis P-PID controller values. Because all of the attitude and altitude controllers were not calibrated, the UAV could crash just before takeoff. For this reason, the UAV was held by a pair of wires that prevented the device to make pitch axis, yaw axis, lateral and up-down movements. The determined parameters for the roll controller were $K_{p1\phi}$ = 9.5611, $K_{p2\phi}$ = 0.3727, $K_{i\phi}$ = 0.1812 and $K_{d\phi}$ = 0.0064.

After determining good parameters for the roll axis controller and the obtained inertia and propeller-motor parameters, a simulation for the roll axis behavior was made in the Simulink © platform from MATLAB ©. As is shown in Fig. 4, both of the simulation and real UAV roll axis behavior are very similar in the rise time.

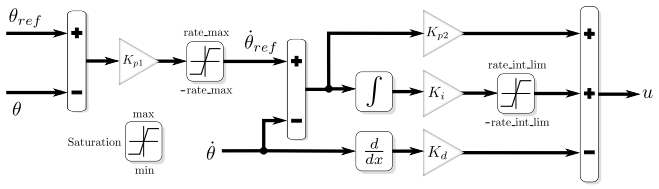

Fig. 3. Structure of the PX4 attitude controller [6].

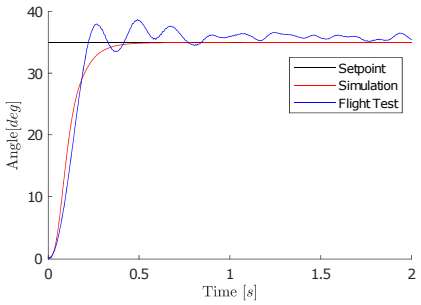

Fig. 4. UAV movement in the roll axis comparison.

III. Control Parameters Optimization

A. Pareto dominance

In a multi-objective optimization problem D objectives $y_i = f_i(\mathbf{x})$, where $i = 1,\cdots,D$; have to be optimized, and each objective depends of a vector $\mathbf{x}$ of K parameters of decision variables. A decision vector $\mathbf{u}$ is said to *strictly dominate* another vector $\mathbf{v}$ if $f_i(\mathbf{u}) < f_i(\mathbf{v})$ $\forall i = 1,2,3,\cdots,D$; furthermore $\mathbf{u}$ *weakly dominates* $\mathbf{v}$ if $f_i(\mathbf{u}) \leq f_i(\mathbf{v})$ $\forall_i = 1,\cdots,D$.

The locus of the non-dominated vectors relative to a specific problem is known as the *Pareto Front* of said problem, these vectors are known as *Pareto Optimals*.

B. PSO description

The PSO algorithm is a stochastic metaheuristic method proposed in [14]. It was inspired in the social behavior of animals, where has been noticed an influence of the behavior of a swarm on the individuals of such. These interactions offer a benefit to the search capability of every member of the swarm and give, in general, good control of the trade-off between convergence speed and exploration capability. In the PSO algorithm, each particle $\mathbf{x}_n$ in the swarm of N particles has a velocity of $\mathbf{v}_n$, which determines its location in the next iteration according to:

$$\mathbf{x}_n^{(t+1)} = \mathbf{x}_n^{(t)} + \chi \mathbf{v}_n^{(t)} + \epsilon^{(t)} \tag{10}$$

where $\chi \in [0,1]$ is a constraint value that limits the magnitude of the velocity of every particle and ϵ is a stochastic vector in charge of increasing the exploitability of the swarm, and it is known as the *turbulence factor*. The velocity $\mathbf{v}_n$ of each particle is modified so that these fly towards their personal best $\mathbf{P}_n$, in order to exploit the best results obtained by the particle until that moment, and the global best $\mathbf{G}$, the best particle found by the whole swarm, to obtain an exchange of information between the particles. The concept of optimization (best) is established by a fitness function $f(x_n)$. All of this is accomplished by updating the velocity vector in the following way:

$$\mathbf{v}_n^{(t+1)} = w\mathbf{v}_n^{(t)} + r_1 c_1(\mathbf{P}_n - \mathbf{x}_n^{(t)}) + r_2 c_2(\mathbf{G} - \mathbf{x}_n^{(t)}) \tag{11}$$

where r_1 and r_2 are random uniformly distributed numbers in the range of [0 1], c_1 and c_2 are control values

for establishing the influence of the global and personal bests. Finally, w is a faction known as $inertia$, which controls the trade-off relationship between convergence and exploitability.

C. MOPSO description

The main difficulty in extending the PSO to multiple objective problems is the choice of the best guides for the swarm. One approach is to establish a single cost function equal to a weighted sum of the different objectives to be achieved [9]. In these cases, there is very little control over the trade-off relationship between the different objectives and there are great possibilities of falling to local minimums. Therefore, as with other multi-purpose algorithms, the concept of Pareto Optimals is used as the fitness function for the algorithm, so that the user can choose a desirable solution from the Pareto Front thus ensuring that it is optimal. This has shown good results in previous works [10], [15], [16]. As a consequence, that method was chosen for the present work. A repository $\mathbf{A}$ of non-dominated particles constitutes the Pareto front, and this repository is consulted for the selection of global guides for each particle, using the "SHR" and "PROB" methods described in [10] for keeping the particles in the search space and for selecting the best global guides respectively. In SHR, assuming that the k-th component of a particle $\mathbf{x}_n$ exceeds its corresponding boundary U, the magnitude of $\mathbf{v}_n$ is shrunk according to:

$$\mathbf{x}_n^{(t+1)} = \mathbf{x}_n^{(t)} + \sigma(\chi \mathbf{v}_n^{(t)} + \epsilon^{(t)}) \tag{12}$$

with

$$\sigma = \frac{\mathbf{x}_{nk}^{(t)} - U}{\chi \mathbf{v}_{nk}^{(t)} + \epsilon_k} \tag{13}$$

so that the particle arrives exactly at the limit of the search space. In PROB, for each particle x_n a global guide is chosen among the particles of $\mathbf{A}$ that dominate x_n. Then, the guide is chosen randomly with a probability function proportional to the inverse of the number of particles of the swarm that are dominated currently by those particles. In case that x_n belongs to $\mathbf{A}$, a particle of $\mathbf{A}$ is chosen randomly with the same probability function as before.

D. Optimization procedure

To evaluate the tuning performance of the proposed algorithm, the UAV and its controller were simulated by using its model and the controller shown in Fig. 3. In order to run the MOPSO algorithm, the script needs the following three inputs: (1) the mass $\mathbf{m}$ of the UAV, (2) the moment of inertia $\mathbf{I}$ of the UAV, and (3) the torque and thrust responses of the propellers. Furthermore, the number of particles N, the number of generations G, the control factors c_1, c_2 and w, the velocity constraint value χ and the boundaries of the search space U have to be defined as well.

The UAV starts from standstill in every degree of freedom. Then, the MOPSO algorithm analyzes the step response of the UAV. The proposed PID tuning procedure is run once for all four axes to obtain its corresponding tuned parameters. The parameter search program can be summarized by the pseudocode shown in Algorithm 1, where X is the set of all the particles whose number is defined beforehand, and V are the correspondent velocities of these particles. After the parameter search is ended, the program will display the choiced parameters according to the pre-defined criteria. The values so obtained are then manually loaded to the UAV through programs such Mission Planner© or QGroundControl© open-source software suites.

Algorithm 1 Proposed MOPSO algorithm.

$G, N, c_1, c_2, w, \chi \leftarrow define$ {Assign values to the control factors.}
$X, V, \mathbf{Pb}_n, \mathbf{Gb} \leftarrow initialize()$ {Randomly initialize particles and their velocities}
$\mathbf{A} \leftarrow \emptyset$ {Initially empty archive}
while $t \leq G$ **do**
  while $n \leq N$ **do**
    $\epsilon \leftarrow random$ {Update turbulence factor}
    Update $\mathbf{v}_n^{(t)}$ with (11) and $\mathbf{x}_n^{(t)}$ with (10).
    if $\mathbf{x}_n^{(t)}$ exceeds a boundary **then**
      Enforce constraints with (12) and (13)
    end if
    $[K_{p1\phi}, K_{p2\phi}, K_{i\phi}, K_{d\phi}, K_{p1\theta}, K_{p2\theta}, K_{i\theta}, K_{d\theta}, ...$
    $K_{p1\psi}, K_{p2\psi}, K_{i\psi}, K_{d\psi}, K_{p1pd}, K_{p2pd}, K_{ipd}, ...$
    $K_{dpd}] \leftarrow \mathbf{x}_n^{(t)}$ {Use the position of the particle as the parameters for a new PID controller.}
    Simulate UAV with its new controller's gains.
    $\mathbf{x}_n^{(t)} \leftarrow [O_\phi, RMSE_\phi, RT_\phi, O_\theta, RMSE_\theta, RT_\theta, ...$
    $O_\psi, RMSE_\psi, RT_\psi, O_{pd}, RMSE_{pd}, RT_{pd}]$
    {Overshoot (O), root-mean-square error (RMSE) and rise-time (RT) are used as objectives, these are obtained from the UAV simulation}
    if $u \npreceq \mathbf{x}_n^{(t)} \forall u \in A$ **then**
      $\mathbf{A} \leftarrow u \in \mathbf{A} \mid \mathbf{x}_n^{(t)} \nprec u$ {Remove particles dominated by $\mathbf{x}_n^{(t)}$ from $\mathbf{A}$}
      $\mathbf{A} \leftarrow \mathbf{A} \cup \mathbf{x}_n^{(t)}$ {Add $\mathbf{x}_n^{(t)}$ to $\mathbf{A}$}
    end if
    if $\mathbf{x}_n^{(t)} \preceq \mathbf{Pb}_n \vee (\mathbf{x}_n^{(t)} \nprec \mathbf{Pb}_n \wedge \mathbf{Pb}_n \nprec \mathbf{x}_n^{(t)})$ **then**
      $\mathbf{Pb}_n \leftarrow \mathbf{x}_n^{(t)}$ {Update personal guide}
    end if
    $\mathbf{Gb} \leftarrow \mathbf{A}_j$ {Update global guide for the next particle. Here, j is an index from $\mathbf{A}$ chosen randomly using PROB (see Subsection III-C).}
    $n := n + 1$
  end while
  $t := t + 1$
end while
Select one of the particles from $\mathbf{A}$ as the final tuning.

IV. Obtained results

The platform Matlab-Simulink was used to simulate the UAV-PX4 system and implement the MOPSO algorithm, using as particles the gains $0 \leq K_{p1\phi} \leq 12$, $0 \leq K_{p2\phi} \leq 0.5$, $0 \leq K_{i\phi} \leq 0.2$, $0 \leq K_{d\phi} \leq 0.01$, $0 \leq K_{p1\theta} \leq 12$, $0 \leq K_{p2\theta} \leq 0.6$, $0 \leq K_{i\theta} \leq 0.2$, $0 \leq K_{d\theta} \leq 0.01$, $0 \leq K_{p1\psi} \leq 5$, $0 \leq K_{p2\psi} \leq 0.6$, $0 \leq K_{i\psi} \leq 0.2$, $0 \leq K_{d\psi} \leq 0.1$, $0 \leq K_{p1pd} \leq 1.5$, $0.1 \leq K_{p2pd} \leq 0.4$, $0.01 \leq K_{ipd} \leq 0.1$ and $0 \leq K_{dpd} \leq 0.1$ of the attitude and altitude controllers, thus having a 16-dimensional search space for the particle swarm. The values of the boundaries are found in [17]. As an objective vector, the overshoot (OS), rise time (RT) and root-mean-square error (RMSE) were used for each degree of freedom, with a total of 12 dimensions for the solution space, where the Pareto front can be visualized.

The number of particles used in the present work was 24, with 90 generations (iterations). Furthermore, w = 0.5, c_1=1, c_2=0.9 and χ=1 were used as search parameters. All the mentioned PSO parameters were empirically chosen. To avoid local minimums, in order to give place to occasional big leaps for the particles, a perturbation using a Laplacian distribution:

$$p(\epsilon_k) \propto e^{(-|\epsilon_k|)/\beta},$$

with $\beta = 0.1$ has been used to define the components of the turbulence factor [10]. For this work, 4 calibration tests were made to test the repeatability and reliability of the prposed algorithm. Figs. 5-7 show the obtained Pareto front for the 4 axes. Note that the figures are related to OS, RT and RMSE. The size of the particle represents the value of control performance on the altitude axis.

For each pareto front, the particle with the smallest value of the norm of the vector [RMSE$_\phi$, RMSE$_\theta$, RMSE$_\psi$, RMSE$_{pd}$] was selected. The behavior of the flight controller was simulated for each particle seletected from the callibration tests and its results are shown in Figs. 8-11 and Table I. The four particles show a very good behavior in all axes. The average computation time was 45 min. using a computer with a Intel® Core™ i7-4720HQ processor and 16 GB of DDR3-1600 RAM.

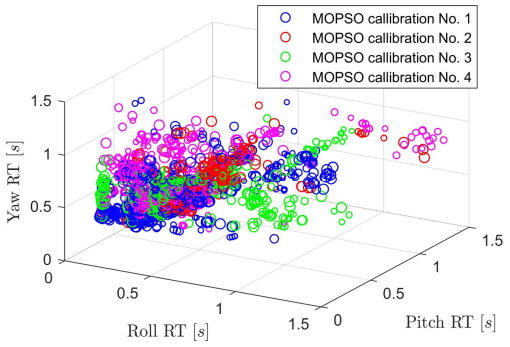

Fig. 5. Pareto front of the rise time of the 4 axis controllers.

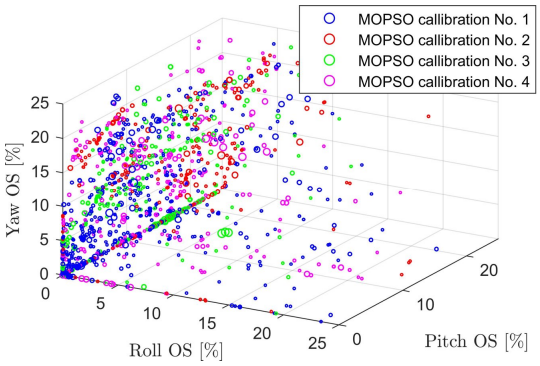

Fig. 6. Pareto front of the overshoot of the 4 axis controllers.

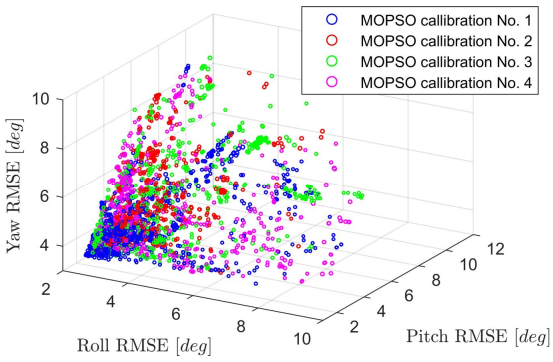

Fig. 7. Pareto front of the root-mean-square error of the 4 axis controllers.

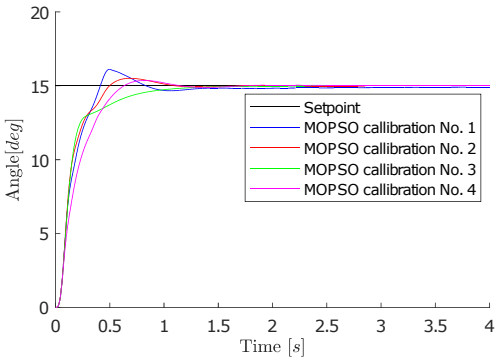

Fig. 8. Roll axis step response for the 4 MOPSO callibration tests.

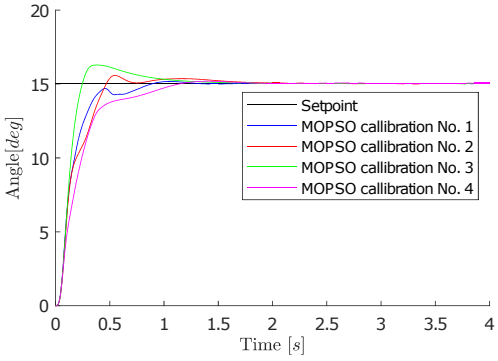

Fig. 9. Pitch axis step response for the 4 MOPSO callibration tests.

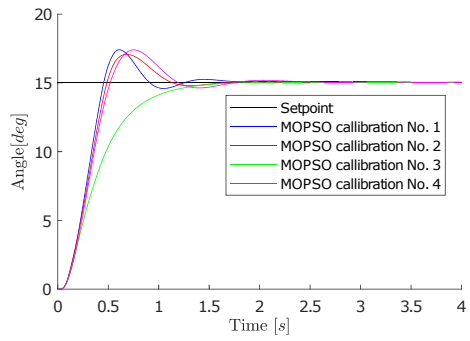

Fig. 10. Yaw axis step response for the 4 MOPSO callibration tests.

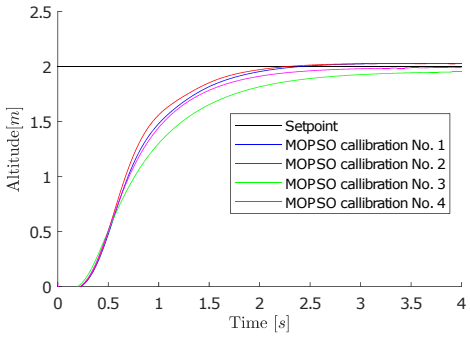

Fig. 11. Altitude axis step response for the 4 MOPSO callibration tests.

TABLE I

OBTAINED CONTROL PERFORMANCE BASED ON SIMULATION RESULTS

Performance value	Test 1	Test 2	Test 3	Test 4
OS_{ϕ}	0.87 %	3.35 %	0.01 %	1.60 %
RT_{ϕ}	0.37 s	0.33 s	0.40 s	0.47 s
$RMSE_{\phi}$	2.40 deg	2.32 deg	2.38 deg	2.65 deg
OS_{θ}	0.88 %	0.36 %	7.45 %	1.18 %
RT_{θ}	0.32 s	0.33 s	0.20 s	0.45 s
$RMSE_{\theta}$	2.33 deg	2.42 deg	2.22 deg	2.65 deg
OS_{ψ}	14.85 %	14.45 %	0.04 %	15.78 %
RT_{ψ}	0.41 s	0.42 s	0.83 s	0.45 s
$RMSE_{\psi}$	3.34 deg	3.48 deg	3.83 deg	3.57 deg
OS_{pd}	0.01 %	0.01 %	0.01 %	0.01 %
RT_{pd}	1.50 s	1.39 s	1.90 s	1.52 s
$RMSE_{pd}$	0.68 m	0.66 m	0.73 m	0.70 m

V. CONCLUSIONS

In this paper, an offline tuning procedure for a commercial Px4-based UAV based on particle swarm optimization has been introduced. Since Px4-based UAV needs to be calibrated before it can be used, a technique based on MOPSO has been proposed. The simulation results show good performance considering the optimization of the overshoot, rise time and root-mean-square error of step response of the P-PID controllers. Note that the proposed method can be easily extended to other multirotor configurations (i.e. hexacopter, octocopter, etc). A possible direction for future work could be the online experimental implementation of the MOPSO.

ACKNOWLEDGEMENTS

The authors would like to thank to the Consejo Nacional de Ciencia y Tecnología (CONACYT) research project (PINV15-0136).

REFERENCES

[1] H. Menouar, I. Guvenc, K. Akkaya, A. S. Uluagac, A. Kadri, and A. Tuncer, "UAV-enabled intelligent transportation systems for the smart city: Applications and challenges," *IEEE Communications Magazine*, vol. 55, pp. 22–28, Mar. 2017 doi: 10.1109/MCOM.2017.1600238CM.

[2] Y. Kali, J. Rodas, R. Gregor, M. Saad, and K. Benjelloun, "Attitude tracking of a tri-rotor UAV based on robust sliding mode with time delay estimation," in *Proc. ICUAS*, 2018, doi: 10.1109/ICUAS.2018.8453472.

[3] Y. Kali, J. Rodas, M. Saad, K. Benjelloun, M. Ayala, and R. Gregor, "Finite-time altitude and attitude tracking of a tri-rotor UAV using modified super-twisting second order sliding mode," in *Proc. ICINCO*, 2018, pp. 445-452, doi: 10.5220/0006861904450452.

[4] E. Paiva, J. Rodas, Y. Kali, R. Gregor, and M. Saad, "Robust flight control of a tri-rotor UAV based on modified super-twisting algorithm," in *Proc. ICUAS*, 2019, pp. 551-556, doi: 10.1109/ICUAS.2019.8797742.

[5] E. Paiva, M. Gómez, J. Rodas, Y. Kali, M. Saad, and R. Gregor, "Cascade first and second order sliding mode controller of a quadrotor UAV based on exponential reaching law and modified super-twisting algorithm," in *Proc. RED UAS*, 2019, pp. 100-105, doi: 10.1109/REDUAS47371.2019.89.

[6] N. Ortiz, E. Laroche, R. Kiefer, and S. Durand, "Controller tuning strategy for quadrotor MAV carrying a cable-suspended load," in *Proc. IMAV*, 2016, pp. 1051-1056, doi: 10.1109/CEC.2002.1004388.

[7] D. Brescianini, M. Hehn, and R. D'Andrea, "Nonlinear quadrocopter attitude control," ETHZ, Zurich, Switzerland, Accessed: Jan. 16, 2020. [Online]. Available: https://doi.org/10.3929/ethz-a-009970340.

[8] "Px4 tuning guide." https://docs.px4.io/v1.9.0/en/config_mc/pid_tuning_guide_multicopter.html (accessed Jan. 16, 2020).

[9] T. Mac, C. Copot, T. Duc, and R. De Keyser, "AR. drone UAV control MOPSO tuning based on particle swarm optimization algorithm," in *Proc. AQTR*, 2016, pp. 1-6, doi: 10.1109/AQTR.2016.7501380.

[10] J. E. Alvarez-Benitez, R. M. Everson, and J. Fieldsend, "A MOPSO algorithm based exclusively on pareto dominance concepts," in *Proc. EMO*, 2005, pp. 459-473, doi: 10.1007/978-3-540-31880-4_32.

[11] E. Paiva, M. Llano, J. Rodas, R. Gregor, J. Rodríguez-Piñeiro, and M. Gómez, "Design and implementation of a VTOL flight transition mechanism and development of a mathematical model for a tilt rotor UAV," in *Proc. ICA-ACCA*, 2018, pp. 1-6, doi: 10.1109/ICA-ACCA.2018.8609836.

[12] R. Beard and T. McLain, *Small unmanned aircraft: Theory and practice*. Princeton, NJ, USA: PU Press, 2012.

[13] M. Krznar, D. Kotarski, P. Piljek, and D. Pavkovic, "On-line inertia measurement of unmanned aerial vehicles using on board sensors and bifilar pendulum," *INDECS*, vol. 16, pp. 149–161, Jan. 2018.

[14] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. ICNN*, 1995, pp. 1942-1948, doi: 10.1109/ICNN.1995.488968.

[15] C. Coello and M. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," in *Proc. CEC*, 2002, pp. 1051-1056, doi: 10.1109/CEC.2002.1004388.

[16] J. E. Fieldsend and S. Singh, "A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence," in *Proc. UKCI*, 2002, pp. 37 - 44, [Online] Available: http://hdl.handle.net/10871/11690.

[17] "Px4 parameters." https://dev.px4.io/v1.9.0/en/advanced/parameter_reference.html (accessed Jan. 16, 2020).