Trajectory planning and control for drone replacement for multidrone cinematography

Marta Marques [1] **Bruno J. Guerreiro** [2] **Rita Cunha** [1]
Carlos Silvestre [3]

Abstract:
This work addresses the problem of trajectory planning and control for a team of drones, so that a vehicle that is following a target as part of a multi-vehicle formation flight can be safely replaced with minimum impact on the formation objectives. An optimal trajectory generation method is presented, which relies on formulating a nonlinear optimization problem with constraints imposed by the physical limitations of the vehicle, the formation, and the surrounding environment, including nonlinear constraints to enforce inter-vehicle collision avoidance and clear visibility of the target. An online planning and control strategy is also implemented through a Nonlinear Model Predictive Control problem, based on the dynamic model of the system, a quadratic objective function, and a set of constraints, that again include collision avoidance and clear visibility. The proposed algorithms are validated in software-in-the-loop simulations that include the autopilot software.

Keywords: UAV dynamics, control, guidance and navigation, Formation flying.

1. INTRODUCTION

Over the past years, there have been significant technological developments in automotive, sensor, and computer industries, which empowered the development of rotary-wing unmanned aerial vehicles, or drones, as highly versatile tools for industrial and consumer applications. Multiple studies have been successful in defining strategies for the motion control of these vehicles in free flight. Nonetheless, new challenges arise when multiple drones are required to share the same airspace or to work together in order to complete a certain task. The motivation for this work arises from the importance of devising methods for the replacement of energy depleted drones during formation flight, in such a way as to minimize the disruption of the formation and tracking, while ensuring the safety of every drone as well as the surrounding structures and people.

When it comes to optimal trajectory planning, research studies differ mostly on the type of trajectories and on the type of constraints used. Nonlinear approaches are considered in Augugliaro et al. (2012), where the problem is solved with a Sequential Convex Programming (SCP) algorithm, whereas other studies are able to replace the nonlinear constraints with mixed-integer linear constraints, which is the case of Mellinger et al. (2012), where optimal polynomial trajectories are generated from a Mixed-Integer Quadratic Problem (MIQP). Regarding real time planning and control strategies, some studies have already used MPC to generate and follow collision free trajectories for multiple drones such as Bemporad and Rocchi (2011), but these do not consider the drone's field of view. The authors in Falanga et al. (2018) are able to solve this issue by considering an MPC with both action and perception objectives, while the method presented in Nageli et al. (2017) generates trajectories in real-time with applications in aerial videography, taking into account visibility under occlusion and collision avoidance.

The main goal of this work is to generate optimal trajectories that are able to solve the problem of drone replacement. In a first stage, we will develop a planning strategy to generate optimal trajectories by solving nonlinear constrained optimization problems. In a second stage, the idea is to evolve from off-board trajectory generation approaches to online planning. In this case, planning and control will be implemented simultaneously and in real-time, relying on concepts of Model Predictive Control (MPC).

The manoeuvres considered in a general scenario with Q drones, where drone i, with $i = 1, ..., Q-1$, is replaced by drone q, are presented as follows. Consider that in the beginning $Q-1$ drones are following the target in their respective positions in formation flight. When drone i needs to be replaced, another drone (drone q) takes off and joins the formation flight with an offset from the formation position of drone i. Then drones i and q

[1] Marta Marques and Rita Cunha are with Institute for Systems and Robotics (ISR-Lisbon), LARSYS, Instituto Superior Técnico, 1049-001 Lisbon, Portugal. (e-mails: marta.marques@tecnico.ulisboa.pt, rita@isr.tecnico.ulisboa.pt)

[2] Bruno J. Guerreiro (corresponding author) is with ISR-Lisbon, and also with the Dep. of Electrical and Computer Eng. and CTS-Uninova, Universidade Nova de Lisboa, Caparica 2829-516, Portugal (e-mail: bj.guerreiro@fct.unl.pt).

[3] Carlos Silvestre is with the Dep. of Electrical and Computer Eng., Faculty of Science and Technology, University of Macau, China, and also on leave from Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal (e-mail: csilvestre@umac.mo).

[4] This work was partially funded by the FCT projects REPLACE (LISBOA-01-0145-FEDER-032107), LOTUS (PTDC/EEIAUT/5048/2014), LARSyS (UID/EEA/50009/2019), and CTS (UID/EEA/00066/2019), by the EU's Horizon 2020 programme (grant No 731667, MULTIDRONE), as well as by the projects MYRG2018-00198-FST of the University of Macau and by the Macau's FDCT under Grant FDCT/026/2017/A1.

exchange positions avoiding interferences with the other drones, and liberating drone i to leave the formation flight and land.

The first contribution of this paper is to implement and adapt a planning method and to solve the problem of drone replacement in formation flight for a multidrone situation by including additional constraints that account for the formation and for the vehicles' field of view during flight. A second contribution relies on the formulation of a Nonlinear Model Predictive Control (NMPC) problem that addresses the same goals but is able to adapt to the target and formation trajectory changes in real time, as demonstrated in realistic software-in-the-loop simulations.

The remainder of this work is structured as follows. In Section 2, we describe a planning strategy to follow a target in formation flight. Section 3 proposes an NMPC algorithm to perform online trajectory planning and control. In Section 4, we validate the planning algorithms with Matlab and Software In The Loop (SITL) simulation results. Finally, Section 5 summarizes the goals achieved in this work and discusses what can still be done as future work.

2. OPTIMAL TRAJECTORY GENERATION

An optimization problem capable of generating collision free discrete trajectories for multiple vehicles is formulated using non-trivial constraints necessary for drone replacement, resulting in a centralized nonlinear optimization problem. The method described in this section builds on the work presented in Augugliaro et al. (2012), adding formation and camera visibility constraints to the optimization problem as presented below.

2.1 Trajectory dynamics

Considering that there are Q vehicles and K time steps, $p_{ij}[k]$, $v_{ij}[k]$ and $a_{ij}[k]$ are the position, velocity and acceleration, respectively, of a vehicle i in the world frame, with $i = 1, ..., Q$, in the direction axis $j \in \{x, y, z\}$ and in the time step $k \in [0, \dots, K]$, and h is the discretization time. Then, we may have the following discrete system dynamics:

$$\begin{aligned} v_{ij}[k+1] &= v_{ij}[k] + h a_{ij}[k], \\ p_{ij}[k+1] &= p_{ij}[k] + h v_{ij}[k] + \frac{h^2}{2} a_{ij}[k]. \end{aligned} \tag{1}$$

The optimization variable, $\boldsymbol{\chi} \in \mathbb{R}^{3QK}$, corresponds to the vehicles' accelerations at each time step in the form

$$\boldsymbol{\chi} = \left[a_{1x}[0], \dots, a_{1x}[K], a_{1y}[0], \dots, a_{1y}[K], \dots, a_{Qz}[0], \dots, a_{Qz}[K]\right]^T.$$

In order to obtain expressions for the position and velocity that only depend on $\boldsymbol{\chi}$, the equations in (1) can be modified as

$$\begin{aligned} v_{ij}[k] &= v_{ij}[0] + h(a_{ij}[0] + a_{ij}[1] + \dots + a_{ij}[k-1]), \\ p_{ij}[k] &= p_{ij}[0] + h(k-1)v_{ij}[0] + \frac{h^2}{2}((2k-3)a_{ij}[0] \\ &\quad + (2k-5)a_{ij}[1] + \dots a_{ij}[k-1]). \end{aligned}$$

If we repeat these equations for all Q vehicles and in the 3 direction axis (x, y and z), we obtain a system of equations that can be written in matrix form, such as

$$\begin{aligned} \mathbf{v} &= H_{11}\boldsymbol{\chi} + \mathbf{v_0}, \\ \mathbf{p} &= \frac{h^2}{2} H_{22}\boldsymbol{\chi} + h H_{33} \odot \mathbf{v_0} + \mathbf{p_0}, \end{aligned}$$

where $\mathbf{v} \in \mathbb{R}^{3QK}$, $\mathbf{p} \in \mathbb{R}^{3QK}$, $H_{11} \in \mathbb{R}^{3QK \times 3QK}$, $H_{22} \in \mathbb{R}^{3QK \times 3QK}$, $H_{33} \in \mathbb{R}^{3QK}$, $\mathbf{v_0} \in \mathbb{R}^{3QK}$, $\mathbf{p_0} \in \mathbb{R}^{3QK}$ and $\odot$ represents an element-wise matrix multiplication.

2.2 Optimization problem

An optimization problem is defined by an optimization variable $\boldsymbol{\chi}$, an objective function f_0, and a set of constraints, and can be generally formulated as:

$$\begin{aligned} &\min \ f_0(\boldsymbol{\chi}) \\ &\text{subject to } A_{eq}\boldsymbol{\chi} = \mathbf{b_{eq}}, \\ &\qquad\qquad A_{in}\boldsymbol{\chi} \leq \mathbf{b_{in}}, \\ &\qquad\qquad c_i(\boldsymbol{\chi}) \leq 0 \end{aligned}$$

where A_{eq} and $\mathbf{b_{eq}}$ are matrices that represent linear equality constraints, A_{in} and $\mathbf{b_{in}}$ represent linear inequality constraints, and c_i denotes nonlinear constraints.

The objective function is defined as the sum of the total mass-normalized thrust at each time step. Considering that $m\mathbf{a} = -mg\mathbf{z_W} + \mathbf{F} \iff \mathbf{F}/m = \mathbf{a} + g\mathbf{z_W}$, where $g = 9.81 m/s^2$ is the gravity constant, $\mathbf{z_W} = [0, 0, 1]^T$ is the z axis in the world frame, $\mathbf{F}$ is the total thrust and m is the mass of the vehicle, then the objective function can be formulated as

$$f_0 = \sum_{i=1}^{Q}\sum_{k=0}^{K} \|\mathbf{a_i}[k] + g\mathbf{z_W}\|^2,$$

with $\mathbf{a_i}[k] = [a_{ix}[k], a_{iy}[k], a_{iz}[k]]^T$, which can be rewritten in the quadratic form $f_0(\boldsymbol{\chi}) = \boldsymbol{\chi}^T P \boldsymbol{\chi} + q^T\boldsymbol{\chi} + r$.

Regarding the constraints, we considered the following linear equalities:

(1) Initial and final states - to define the initial and final positions, velocities and accelerations of each vehicle. Suppose that $k_I = 0, ..., K_I$ are the time indexes for which the initial state of the vehicles remains the same and $k_F = K_F, ..., K$ are the time indexes for which their final state remains the same. Then, for a vehicle i, with $i = 1, ..., Q$, and in the direction axis $j \in \{x, y, z\}$, the constraints are:

$$\begin{aligned} a_{ij}[k_I] &= a_{ij0}, & a_{ij}[k_F] &= a_{ijK}. \\ v_{ij}[k_I] &= v_{ij0}, & v_{ij}[k_F] &= v_{ijK}, \\ p_{ij}[k_I] &= p_{ij0}, & p_{ij}[k_F] &= p_{ijK}. \end{aligned} \tag{2}$$

(2) Formation flight for target following - achieved by assigning specific parameters (ϵ_{ti}, ϵ_{xi}, ϵ_{yi}, and ϵ_{zi}, with $i = 1, ..., Q$) to different positions, as represented in Figure 1, so that the vehicles are distanced in time and space in relation to the target's position and orientation. Consider an additional coordinate system, the target frame, $\mathcal{T}$, and the rotation matrix denoted by ${}^W R_T[k]$ that describes the orientation of the target in the world frame, $\mathcal{W}$, in a time step $k = 0, ..., K$. To ensure that a vehicle i, with $i = 1, ..., Q$, follows the target in the respective position in formation flight, the following constraint must be implemented:

$$\mathbf{p_i}[k] = \mathbf{p_T}\big[k - \frac{\epsilon_{ti}}{h}\big] + {}^W R_T\big[k - \frac{\epsilon_{ti}}{h}\big]\boldsymbol{\epsilon}_{\mathbf{p}i}. \tag{3}$$

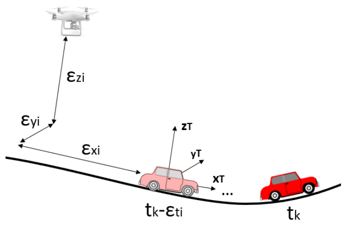


Fig. 1. Position i in formation flight.

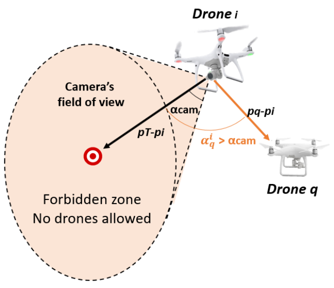


Fig. 2. Illustration of the clear visibility constraint.

where $\boldsymbol{\epsilon}_{\mathbf{p}i} = [\epsilon_{xi}, \epsilon_{yi}, \epsilon_{zi}]^T$ is a vector with the formation positions.

Additionally, we included the following linear inequalities:

(1) Dynamics limitation - to define the maximum and minimum values allowed for the position, velocity, acceleration and jerk. For a vehicle i, with $i = 1, ..., Q$, and in any time step $k = 0, ..., K$, we have the constraints: $\mathbf{p_{min}} \leq \mathbf{p_i}[k] \leq \mathbf{p_{max}}$, $\mathbf{v_{min}} \leq \mathbf{v_i}[k] \leq \mathbf{v_{max}}$, $\mathbf{a_{min}} \leq \mathbf{a_i}[k] \leq \mathbf{a_{max}}$, and $\mathbf{j_{min}} \leq \mathbf{j_i}[k] \leq \mathbf{j_{max}}$, where $\mathbf{j}$ represents the jerk, which measures the variation of acceleration and can be computed as $\mathbf{j}[k] = (\mathbf{a}[k] - \mathbf{a}[k-1])/h$.

Finally, the following nonlinear constraints are considered:

(1) Collision avoidance - imposes that the distance between drones is larger than a certain safety distance d_{saf}. For any drones i and j, with $i = 1, ..., Q$, $j = 1, ..., Q$ and $i \neq j$, and k is any time step in $[0, ..., K]$, this constraint can be formulated as

$$\|\mathbf{p_i}[k] - \mathbf{p_j}[k]\| \geq d_{saf}. \tag{4}$$

(2) Clear visibility during replacement - When drones i and q are exchanging positions, they should not be detected by any of the cameras of the other drones in the formation. Considering the situation where drone q cannot be seen by drone i, we assume that drone i's camera is pointed at the target in the direction of the vector $\mathbf{p_{Ti}} = \mathbf{p_T} - \mathbf{p_i}$ and that it has a field of view of $2\alpha_{cam}$. In order for the camera to not detect drone q, the angle between $\mathbf{p_{Ti}}$ and the vector $\mathbf{p_{qi}} = \mathbf{p_q} - \mathbf{p_i}$ should be greater than α_{cam}, which is equivalent to

$$\mathbf{p_{Ti}}[k]^T \mathbf{p_{qi}}[k] - \|\mathbf{p_{Ti}}[k]\| \|\mathbf{p_{qi}}[k]\| \cos(\alpha_{cam}) \leq 0. \tag{5}$$

This constraint is illustrated in Figure 2 where the camera's field of view is represented as a forbidden zone for any other drone.

2.3 Sequential Convex programming

Working directly with nonlinear constraints can be inefficient as they increase the complexity of the problem. As an alternative, the use of Sequential Convex Programming (SCP) replaces these constraints with linear approximations, where at iteration $s+1$, nonlinear constraints are linearized around the previous solution $\boldsymbol{\chi}^s$ using a first order Taylor expansion. For simplicity of notation, the time index k is dropped hereafter whenever it can be inferred from the context.

To linearize the collision avoidance constraint for each instant $k = 0, ..., K$, we define $\mathbf{p_{ij}}^s = \mathbf{p_i}^s - \mathbf{p_j}^s$ and the approximated linear constraint is obtained by linearizing $\|\mathbf{p_{ij}}\|$ around $\mathbf{p_{ij}}^s$, resulting in the constraint

$$\|\mathbf{p_{ij}}^s\| + \frac{\mathbf{p_{ij}}^s}{\|\mathbf{p_{ij}}^s\|}^T (\mathbf{p_{ij}} - \mathbf{p_{ij}}^s) \geq d_{saf}.$$

As for the clear visibility constraint, we consider that $\mathbf{p_{Ti}}^s = \mathbf{p_T} - \mathbf{p_i}^s$, $\mathbf{p_{qi}}^s = \mathbf{p_q}^s - \mathbf{p_i}^s$ and $f(\mathbf{p_{Ti}}, \mathbf{p_{qi}}) = \mathbf{p_{Ti}}^T \mathbf{p_{qi}} - \|\mathbf{p_{Ti}}\| \|\mathbf{p_{qi}}\| \cos(\alpha_{cam})$. The constraint is obtained by linearizing $f(\mathbf{p_{Ti}}, \mathbf{p_{qi}})$, yielding a quadratic optimization problem, with a quadratic objective function and affine constraints.

In the presented problem, the objective function is convex if $\nabla^2 f(x) \succeq 0$, which is true as P is a symmetric semi-positive definite matrix. As for the constraints, the combination of the linear equalities and inequalities will yield a convex set. The nonlinear constraints are, however, not convex, which means that when using these constraints directly, the optimization problem becomes non-convex and the solution found is only locally optimal. By applying the SCP algorithm, the nonlinear constraints become linear for each iteration and the problem can be solved as a quadratic problem with linear constraints, hence convex in each iteration. Yet the convergence of the algorithm can only be guaranteed locally as in the original problem.

3. ONLINE PLANNING AND CONTROL

In this Section, we address a similar problem with a real-time trajectory planning and control algorithm by implementing a centralized NMPC strategy that is able to simultaneously generate trajectories for a short time horizon and control the system in real time. The proposed method builds on the optimization problem defined above and on the approach presented in Falanga et al. (2018), where the NMPC problem is extended to a multi-vehicle situation and the formulation of the optimization problem is similar to the one implemented in Section 2. In the presented strategy, we are explicitly considering the orientation of the vehicle and the position and orientation of the camera.

3.1 System dynamics

Noting that the system composed by the drone and camera will act as an equality constraint for the NMPC optimization problem this section introduces this model. Consider the three coordinate systems: world frame, $\mathcal{W}$, body frame, $\mathcal{B}$, and camera frame, $\mathcal{C}$, as represented in Figure 3. The transformation between coordinate systems

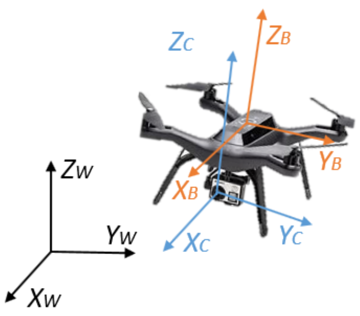


Fig. 3. Representation of the coordinate frames

can be defined by quaternions $\mathbf{q} = q_w + q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k}$, which are also represented by a vector $\mathbf{q}$ in the 3-Sphere $\mathbb{S}^3 = \{\boldsymbol{q} \in \mathbb{R}^4 : \|\boldsymbol{q}\| = 1\}$.

Considering a continuous system dynamics with Q vehicles, the state of the system is given by

$$\mathbf{x} = [\mathbf{p_1}, ..., \mathbf{p_Q}, \mathbf{v_1}, ..., \mathbf{v_Q}, {}^{W}\mathbf{q}_{B1}, ..., {}^{W}\mathbf{q}_{BQ}]^T, \quad (6)$$

where $\mathbf{p_i} = [p_{ix}, p_{iy}, p_{iz}]^T$ and $\mathbf{v_i} = [v_{ix}, v_{iy}, v_{iz}]^T$ represent the position and velocity in the world frame and ${}^{W}\mathbf{q}_{Bi}$ represents the quaternion of the body w.r.t. the world frame of a drone i, with $i = 1, .., Q$.

Regarding the control inputs, we used the mass-normalized thrust $c = \frac{F_1+F_2+F_3+F_4}{m}$, where F_i is the thrust generated by the i^{th} motor, and the angular velocities $\boldsymbol{\omega_B} = [p, q, r]^T$ expressed in the body frame. Thus, the vector of control inputs is given by

$$\mathbf{u} = [c_1, ..., c_Q, \boldsymbol{\omega_{B1}}, ..., \boldsymbol{\omega_{BQ}}]^T. \quad (7)$$

The system dynamics can then be formulated, for each drone i, as

$$\begin{aligned} \dot{\mathbf{p}}_\mathbf{i} &= \mathbf{v_i}, \\ \dot{\mathbf{v}}_\mathbf{i} &= -g\mathbf{z_W} + R_q({}^{W}\mathbf{q}_{Bi})c\mathbf{z_W} \\ {}^{W}\dot{\mathbf{q}}_{iB} &= \frac{1}{2}\Lambda(\boldsymbol{\omega_{Bi}}){}^{W}\mathbf{q}_{Bi}, \end{aligned} \quad (8)$$

where $R_q({}^{W}\mathbf{q}_{Bi})$ is the quaternion rotation operation and $\Lambda(\boldsymbol{\omega_{Bi}})$ is the $[4 \times 4]$ skew-symmetric matrix defined from $\boldsymbol{\omega_{Bi}}$.

3.2 NMPC algorithm

MPC is a method of iteratively solving an optimization problem for a finite time horizon and can be generally formulated as

$$\begin{aligned} &\min_{\mathbf{U}} V_N(\mathbf{X}, \mathbf{U}), \\ &\text{subject to } \mathbf{r}(\mathbf{X}, \mathbf{U}) = 0, \\ &\qquad\qquad \mathbf{h}(\mathbf{X}, \mathbf{U}) \leq 0, \end{aligned}$$

where $\mathbf{X} = [\mathbf{x_0}, ..., \mathbf{x_N}]$ and $\mathbf{U} = [\mathbf{u_0}, ..., \mathbf{u_{N-1}}]$ are state and input sequences for the time horizon, with components defined in (6) and (7), respectively, $V_N(\mathbf{X}, \mathbf{U})$ is the objective function, whereas $\mathbf{r}(\mathbf{X}, \mathbf{U})$ and $\mathbf{h}(\mathbf{X}, \mathbf{U})$ represent the equality and inequality constraints. To solve this optimization problem, it is usually necessary to discretize the system dynamics defined in (8) with a sampling time T_s and to consider a time horizon T_h, so that at each iteration, we obtain a sequence of N inputs $\mathbf{U}$ and a sequence of $N+1$ state predictions $\mathbf{X}$, with $N = T_h/T_s$. The components of the algorithm are described below.

The objective function is formulated as a quadratic function of the state and inputs as

$$\begin{aligned} V_N =& (\mathbf{x_N} - \mathbf{x_N^{ref}})^T \mathcal{Q}_{xN} (\mathbf{x_N} - \mathbf{x_N^{ref}}) + \\ & \sum_{i=0}^{N-1} \Big((\mathbf{x_i} - \mathbf{x_i^{ref}})^T \mathcal{Q}_{xi} (\mathbf{x_i} - \mathbf{x_i^{ref}}) + (\mathbf{u_i} - \mathbf{u_i^{ref}})^T \mathcal{R}_i (\mathbf{u_i} - \mathbf{u_i^{ref}}) \Big), \end{aligned}$$

where:

- $\mathcal{Q}_{xi}$ and $\mathcal{R}_i$, with $i = 0, ..., N-1$, are the state and input cost matrices and $\mathcal{Q}_{xN}$ is the final state cost penalty;
- $\mathbf{x_i^{ref}}$, with $i = 1, ..., N$, and $\mathbf{u_i^{ref}}$, with $i = 0, ..., N-1$, are state and input references that may vary throughout the time horizon.

The state references $\mathbf{x_i^{ref}}$ have the purpose of describing the drones manoeuvres described above, which include taking off and joining formation flight, going to a different position during replacement, then continuing formation flight and finally landing.These reference trajectories will replace the constraints used in Section 2 defined in (2) and (3), as they are defined for each drone and serve as a baseline for the trajectories to be generated by the MPC. As for the input references $\mathbf{u_i^{ref}}$, we can assume that these are zero.

The following constraints are considered in the NMPC problem:

- System dynamics - already expressed in (8). These are discretized using multiple shooting as a transcript method and a Runge Kutta integration scheme, which are executed with *ACADO*'s generation code functions (Ariens et al. (2014)). More details on the ACADO implementation are presented in the next section.
- Control and state bounds - for a vehicle i, with $i = 1, ..., Q$, these can be expressed in the form $-\mathbf{u}_{max} \leq \mathbf{u}_i \leq \mathbf{u}_{max}$ and $\mathbf{x}_{min} \leq \mathbf{x}_i \leq \mathbf{x}_{max}$.
- Collision constraint - similar to (4).
- Clear visibility - similar to (5) but using the angle between the target and the drones w.r.t. the position of its camera. Considering a situation where drone q cannot be seen by drone i, the vector between the position of the target in the world frame, $\mathbf{p_T}$, and the position of drone i's camera is

$$\mathbf{p_{C_iT}} = \mathbf{p_T} - \Big(R_q({}^{W}\mathbf{q}_{Bi})^{\mathbf{Bi}}\mathbf{p_{Ci}} + \mathbf{p_i} \Big),$$

whereas the vector between the position of drone q in the world frame, $\mathbf{p_q}$, and the position of drone i's camera is defined similarly as $\mathbf{p_{C_iq}}$. The constraint imposes that the angle between the vectors $\mathbf{p_{C_iT}}$ and $\mathbf{p_{C_iq}}$ should be greater than α_{cam}, which translates into the following inequality:

$$\mathbf{p_{C_iT}}^T \mathbf{p_{C_iq}} - \|\mathbf{p_{C_iT}}\| \|\mathbf{p_{C_iq}}\| \cos(\alpha_{cam}) \leq 0.$$

To formulate the NMPC problem, we use the *ACADO* toolbox, that generates C/C++ code to solve automatic control and dynamic optimization problems, such as an MPC problem (Ariens et al. (2014)). The algorithm is implemented in a Matlab interface for *ACADO* that allows to formulate the problem in Matlab with a special *ACADO* syntax, which is then solved with a qpOASES solver.

To test the algorithm, one has to define for each iteration the state and input references, the cost matrices $\mathcal{Q}_x$ and $\mathcal{R}$, the bounds for the constraints and the position and velocity of the target for the next N time steps. Then, we can solve the current NMPC problem by using the generated code, which results in sequences of inputs and predicted states. Afterwards, the first input $\mathbf{u_0}$ can be applied to the system dynamics, from which we obtain a new updated state that will then be used in the next MPC iteration. Figure 4 shows a block diagram representative of the MPC controlled system.

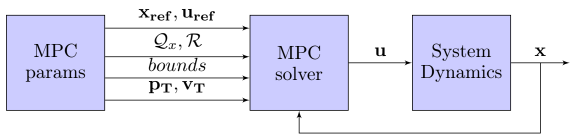

MPC params
$\mathbf{x_{ref}}, \mathbf{u_{ref}}$
$\mathcal{Q}_x, \mathcal{R}$
bounds
$\mathbf{p_T}, \mathbf{v_T}$
MPC solver
u
System Dynamics
x

Fig. 4. Diagram of the MPC controlled system

4. SIMULATION RESULTS

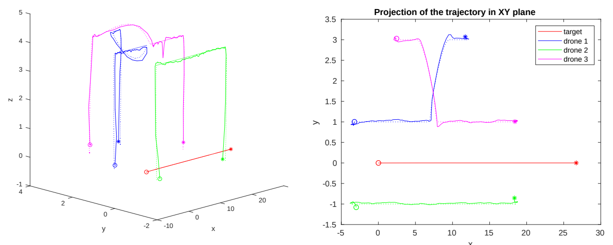
The methods presented in Sections 2 and 3 are validated in this section by presenting realistic simulations both in Matlab and in ROS+Gazebo environment using autopilot in software-in-the-loop (SITL) mode. In the latter, the simulations use the MULTIDRONE simulator (Multidrone (2018)), where the UAVs are setup with a PX4 autopilot software (Meier et al. (2015)). All of the simulations presented in this section were conducted on a Lenovo Laptop equipped with an Intel Core i7-7700HQ CPU @2.80GHz, 16,00GB of RAM, and a NVIDIA GTX 1050 GPU. The Gazebo screen captures of the simulation tests presented in this Section are available in the linked video [5] .

4.1 Tests with pre-planned trajectories

The optimization problem presented in Section 2 is implemented using Matlab's optimization toolbox and *Cplex* (IBM (2015)), yielding optimal waypoint trajectories, that are used as references for the PX4 autopilot in the SITL simulations. We consider a test scenario with 3 drones following a target in a straight line, where drone 1 is replaced by drone 3, and a trajectory of 52 seconds with a fixed time step of $h = 0.5$ seconds.

In the end, we obtain the simulation test showcased in the *Waypoint trajectories video*, where we can see that the drones are able to follow the flight plan previously described without any risk of collision and without being seen by the other drones in formation flight. This can also be verified in the plots from Figure 5, where small deviations are observed between the experimental and reference trajectories and are more predominant during the flight transitions.

To check if the nonlinear constraints are satisfied, the distances between any two drones in the system are also presented in Figure 6, showing that they are always above the minimum safety distance, thus ensuring collision avoidance during the full duration of the test. As for the clear visibility condition, to check if drone 3 is not detected during the time interval dedicated to the replacement process, we compute the angles between drone 3 and the target w.r.t. drone 1 and 2, and then, to check if drone 1 is not detected, we compute the angles between drone 1 and the target w.r.t. drone 2 and 3. These are represented in Figure 7, where we can see that in the Matlab simulation they are always above the camera's field of view angle, but, in the SITL results, this condition is sometimes not verified, even though they are not visible in each drone video stream. This is the result of defining a larger field of view (FoV) in the visibility constraint than the actual camera FoV, thus compensating of disturbances when the drones follow the planned trajectories.



(a) Trajectories in 3D (b) Projection of the trajectories in the plane XY

Fig. 5. Trajectories obtained with the pre-planned trajectories. Note that the *doted line* is the trajectory generated in Matlab, the *full line* is the one in the SITL simulation, and the *circles* and *stars* are the initial and final positions.

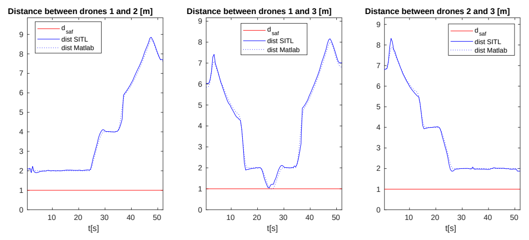


Fig. 6. Distance between drones obtained with the pre-planned trajectory.

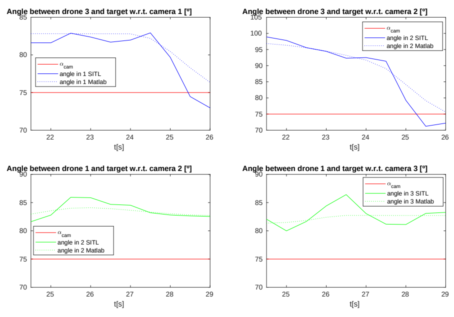
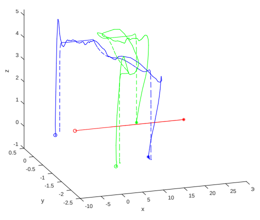


Fig. 7. Clear visibility constraint obtained with the pre-planned trajectory.

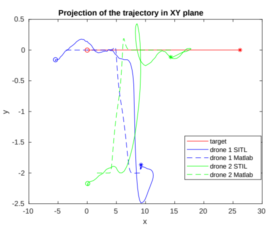
4.2 Tests with online planned trajectories

The MPC algorithm is also validated in a scenario with 2 drones following a similar flight plan as in the above tests, using both Matlab and SITL simulations. For the

[5] https://fenix.tecnico.ulisboa.pt/homepage/ist178289/trajectory-planning-and-control-for-drone-replacement-during-formation-flight

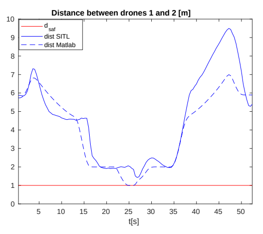


(a) Trajectories in 3D

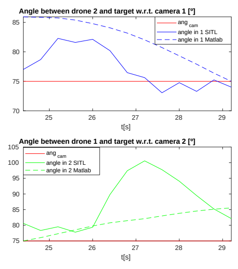


(b) Projection of the trajectories in the plane XY

Fig. 8. Trajectories obtained in the MPC simulation. Note that *circles* and *stars* are the initial and final positions.



(a) Collision avoidance.



(b) Clear visibility.

Fig. 9. Nonlinear constraints in the MPC simulations

latter, we used the linear velocity predictions given by the MPC as control inputs for the PX4 autopilot instead of c and $\boldsymbol{\omega_B}$ used in the Matlab simulation. We will present a test made for a scenario with 2 drones following a similar flight plan as before. We consider here a sampling time of $T_s = 0.4$ seconds and a time horizon of $T_h = 8$ seconds, resulting in sequences of $N = 20$ elements. Note that this is an outer-loop for trajectory tracking, where the inner-loop for position control runs at a much higher rate.

The results of this simulation can be seen in the *MPC trajectories video*. From this, we can see that the proposed algorithm is both effective and implementable, as feasible trajectories are computed in real time. The vehicles manage to execute the necessary manoeuvres and to exchange positions without colliding or being seen. The corresponding trajectories are plotted in Figure 8, where we can see that the SITL trajectories are not as smooth as the ones computed in Matlab and they do not follow the MPC's initial reference as accurately.

Furthermore, we verify if the constraints for collision avoidance and clear visibility are satisfied by plotting the distances and the angles in Figure 9 and note that while collision avoidance is verified in both simulations, clear visibility sometimes fails in the SITL simulation even though this is practically verified in the video as the camera in each drone does not see the other drone. As for the computation time, we obtained an average (maximum) of 60 ms (75 ms) in Matlab and 150 ms (300 ms) in the SITL results, where the higher CPU times are obtained during the exchange manoeuvre. As these CPU times are lower than the sampling time T_s (400 ms), the presented results are a good indicator that the proposed controller can achieve effective drone replacement manoeuvres in a real-time implementation.

5. CONCLUSIONS

The goal of this work was to design a trajectory planning and control strategy to solve the problem of replacement of drones that are following a target in formation flight. In a first stage, we formulated a nonlinear optimization problem with collision avoidance and clear visibility constraints to generate optimal trajectories. These were later validated with simulation tests, where we saw that the drones were able to execute the manoeuvres initially planned without any risk of collision and without being seen during the replacement process. In a second stage, we implemented an NMPC algorithm, taking into account the dynamic model of the system as well as a set of constraints to achieve collision avoidance and clear visibility. We also tested the proposed methods in Matlab and in SITL simulations, validating the real-time implementation of the algorithms.

In terms of future work, there are still several lines of research that could be pursued, such as finding alternative convex constraints for the nonlinear collision avoidance and clear visibility constraints or exploring decentralized MPC approaches. In terms of experimental results, further work is being carried out to test the scenarios presented in Section 4 in real flight experiments.

REFERENCES

Ariens, D., Houska, B., Ferreau, H., and Logist, F. (2014). ACADO Toolkit User's Manual, V 1.2.1beta. `http://www.acadotoolkit.org/`.

Augugliaro, F., Schoellig, A.P., and D'Andrea, R. (2012). Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. *IEEE International Conference on Intelligent Robots and Systems*, 1917–1922.

Bemporad, A. and Rocchi, C. (2011). Decentralized hybrid model predictive control of a formation of unmanned aerial vehicles. *8th IFAC World Congress*.

Falanga, D., Foehn, P., Lu, P., Peng, and Scaramuzza, D. (2018). PAMPC: Perception-Aware Model Predictive Control for Quadrotors. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

IBM (2015). IBM ILOG CPLEX Optimization Studio Getting Started with CPLEX for MATLAB V12.6.

Meier, L., Honegger, D., and Pollefeys, M. (2015). PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. *2015 IEEE International Conference on Robotics and Automation (ICRA 2015)*.

Mellinger, D., Kushleyevand, A., and Kumar, V. (2012). Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. *IEEE International Conference on Robotics and Automation*, 477–483.

Multidrone (2018). Multiple drone platform for media production - multidrone. `https://multidrone.eu/`, Accessed: 2018-09-30.

Nageli, T., Alonso-Mora, J., A. Domahidi, D.R., and Hilliges, O. (2017). Real-Time Motion Planning for Aerial Videography With Dynamic Obstacle Avoidance and Viewpoint Optimization. *IEEE Robotics and Automation Letters*, 2, 1696–1703.