Trajectory planning and control for drone replacement during formation flight

Marta Isabel da Silva Marques
marta.marques@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

November 2018

Abstract

This work addresses the design of trajectory planning and control techniques for a team of drones, so that a vehicle in formation flight can be safely and efficiently replaced. In a first stage, an optimal trajectory generation method is presented, where a nonlinear optimization problem is formulated taking into account constraints imposed by the physical limitations of the vehicle, the formation and the surrounding environment to generate discrete trajectories. Additionally, nonlinear constraints are included to impose collision avoidance and clear visibility during the exchange of positions in replacement. To solve the nonlinear optimization problem, an iterative algorithm based on Sequential Convex Programming is implemented, thus, solving an approximate convex problem at each iteration. Furthermore, a tracking controller is proposed to follow the generated reference trajectories, that is composed by an inner loop responsible for attitude stabilization and an outer loop related to position tracking. In a second stage, an online planning and control strategy is implemented, that is able to generate optimal positions and control inputs in real time. To this end, a Nonlinear Model Predictive Controller is formulated, 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. All of the algorithms presented are validated in simulations that include the autopilot software.
Keywords: Drone, Formation flight, Replacement, Optimal trajectory, Tracking control, Model Predictive Control

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 may occur when multiple drones are required to share the same airspace or to work together in order to complete a certain task. In these cases, it is necessary to not only control each drone individually, but also to study how they interact in the overall task.

The motivation for this work arises from the importance of devising strategies for the replacement of energy depleted drones during formation flight, in such a way as to minimize disruption of the formation and tracking, while ensuring the safety of every drone as well as the surrounding structures and people.

1.1. Objectives

There are two main goals for this work: (i) generate optimal trajectories that are able to solve the problem of drone replacement, and (ii) implement a controller capable of tracking such trajectories.

In a first stage, we will develop planning strategies to generate optimal trajectories by solving linear and/or nonlinear constrained optimization problems, and then implement a controller to track such trajectories that takes into account the underactuated model of a UAV.

Afterwards, in a second stage, the idea is to evolve from offfboard 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).

1.2. Problem Statement

Since the replacement manoeuvres are central to the problem definitions, further details on the type of manoeuvres considered throughout this work are presented in this Section. Consider a scenario where one or more drones are monitoring a target that is following a path. When one of the drones begins to run out of battery, it should be replaced by another one. This means that a new drone must take off and join the formation flight. After a while, they will exchange positions, while guaranteeing that the replacement does not interfere with the overall task. Only then will the drone with low battery be allowed to exit the task and land in an intermediate position. A flight plan for a general scenario with Q drones is presented in Figure 1, where drone i, with $1 = 1, ..., Q - 1$, is replaced by drone q.

This work aims to solve an optimization problem that generates trajectories capable of executing the manoeuvres described while considering the UAV's physical limitations and ensuring collision avoidance between vehicles, which will, then, be used as references in a tracking controller. Afterwards, in a second stage, the problem will be reformulated as an MPC algorithm that handles planning and control simultaneously and in real-time.

More specifically, the contribution of this work will be to implement and adapt a planning method and, later an

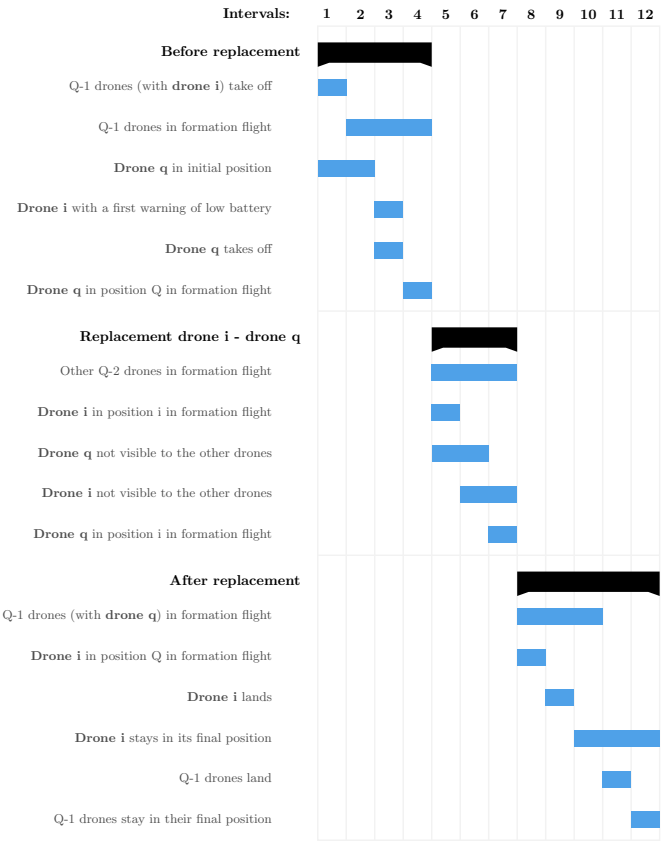


Figure 1: Diagram of a flight plan with Q drones

MPC strategy, 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.

1.3. Structure of the document

The remainder of this work is structured as follows. In Section 2, we mention other studies that have been conducted in the same research field. In Section 3, a mathematical model is presented for a quadrotor, as well as a complementary tracking controller to follow reference trajectories. Section 4 describes a strategy to generate optimal trajectories that are able follow a target in formation flight. Then, in Section 5 a Nonlinear Model Predictive Control (NMPC) algorithm is proposed to perform online trajectory planning and control. In Section 6, we validate the planning algorithms with Matlab and Software In The Loop (SITL) simulations. Finally, Section 7 summarises the goals achieved in this work and discusses what can still be done as future work.

2. Related work

When it comes to optimal trajectory planning, research studies differ mostly on the type of trajectories generated (discrete, polynominal, etc) and on the types of constraints used (nonlinear, with mixed integers, linear, etc). Nonlinear approaches are considered in [1], where the problem is solved with a Sequential Convex Programming (SCP) algorithm and in [2], where an Interior Point algorithm is implemented. Other studies are able to replace the nonlinear constraints with mixed-integer linear constraints, which is the case of [3], where optimal polynomial trajectories are generated from a Mixed-Integer Quadratic Problem (MIQP), and the case of [4], where a Mixed-Integer Linear Problem (MILP) is used to obtain 2D trajectories with constant altitude. Note that even though most of the referenced studies may already account for multidrone trajectory generation and include collision avoidance constraints, they do not explicitly solve the problem of drone replacement in formation flight, nor do they consider restrictions related to the camera's visibility.

As for modelling and trajectory tracking control for quadrotors, the authors in [5] propose a geometric tracking controller, in which four inputs are used to track position and heading. A similar controller is also proposed in [6], where the system is characterized with differentially flat parameters and inside the control loop, errors in position, velocity, rotation and angular velocity are computed at each time-step. Other studies implement MPC techniques to track reference trajectories [7, 8, 9], which can be used as a transition point to online planning and control.

Regarding real time planning and control strategies, some studies have already used MPC to generate and follow collision free trajectories for multiple drones [10, 11], but these do not consider the drone's field of view. Then, there are studies that deal with the vehicle's perception, but only consider situations with one vehicle. Some examples are [12] that uses a minimum time trajectory planning method, [13] that proposes a vision-based navigation method and [14] that suggests a hybrid visual servoing technique. These methods, however, do not take into account the motion of the vehicle. The authors in [15] are able to solve this issue by considering an MPC with both action and perception objectives. Another method is presented in [16], that generates trajectories in real-time with applications in aerial videography, taking into account visibility under occlusion and collision avoidance.

3. Modelling and Control

Consider the two coordinate systems, world frame, $\mathcal{W}$, and body frame, $\mathcal{B}$, represented in Figure 2. The rotation from world to body frame is denoted by the rotation matrix ${}^{W}R_{B}$, which can be parameterized by the roll, pitch and yaw angles, respectively denoted as ϕ, θ, and ψ, resulting in

$$
\begin{aligned}
{}^{W}R_{B} &= R_{\psi}R_{\phi}R_{\theta} \\
&= \begin{bmatrix} c(\psi)c(\theta) - s(\psi)s(\phi)s(\theta) & -s(\psi)c(\phi) & c(\psi)s(\theta) + s(\psi)s(\phi)c(\theta) \\ s(\psi)c(\theta) + c(\psi)s(\phi)s(\theta) & c(\psi)c(\phi) & s(\psi)s(\theta) - c(\psi)s(\phi)c(\theta) \\ -c(\phi)s(\theta) & s(\phi) & c(\phi)c(\theta) \end{bmatrix},
\end{aligned} \tag{1}
$$

according to the $Z - X - Y$ convention, where c and s represent the cosine and sine functions. The angular velocity of the body is denoted by $\boldsymbol{\omega_B} = [p, q, r]^T$ and can be computed according to

$$
\boldsymbol{\omega_B} = \begin{bmatrix} c(\theta) & 0 & -c(\phi)s(\theta) \\ 0 & 1 & s(\theta) \\ s(\theta) & 0 & c(\phi)c(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}, \tag{2}
$$

where $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$ are the derivatives of the roll, pitch and yaw angles.

Each rotor has an angular speed ω_i and produces a force F_i and moment M_i, approximately given by

$$
F_i = k_F \omega_i^2, \qquad M_i = k_M \omega_i^2, \tag{3}
$$

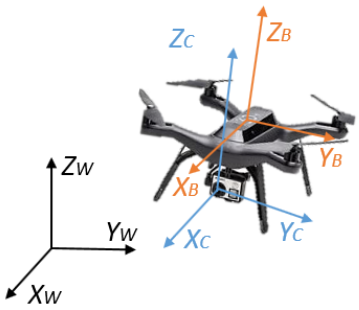


Figure 2: Representation of the coordinate frames

where k_F and k_M are system constants that account for the relation between the speed of the motors and the generated lift and drag forces on each propeller. These can be used to compute the four controller inputs $\mathbf{u}$: u_1, that represents the sum of the forces of the 4 rotors, and $\mathbf{u}_\tau = [u_2, u_3, u_4]^T$, that denotes the body moments in each body axis. If we consider that $\mathbf{u}'$ represents the control inputs for a body frame $\mathcal{B}'$ where the x and y axis are aligned with the quadrotors arms, then the relation between $\mathbf{u}'$ and the rotor speeds ω_i can be written as

$$\mathbf{u}' = \begin{bmatrix} k_F & k_F & k_F & k_F \\ 0 & k_F L & 0 & -k_F L \\ -k_F L & 0 & k_F L & 0 \\ k_M & -k_M & k_M & -k_M \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}, \tag{4}$$

where L is the distance between the axis of the rotors and the center of the quadrotor. The control inputs $\mathbf{u}$ are then given by $\mathbf{u} = {}^B R_{B'} \mathbf{u}'$, where ${}^B R_{B'}$ represents the rotation from $\mathcal{B}'$ to $\mathcal{B}$. Additionally, the state of the system is given by

$$\mathbf{x} = [\mathbf{p}, \mathbf{v}, \boldsymbol{\omega_B}, {}^W R_B]^T, \tag{5}$$

where $\mathbf{p}$ and $\mathbf{v}$ represent the position and velocity of the body frame described in the world frame, respectively.

Furthermore, the system dynamics can be written as

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= -g\mathbf{z_W} + \frac{u_1}{m}\mathbf{z_B}, \\ \dot{\boldsymbol{\omega}}_{\boldsymbol{B}} &= \mathcal{I}^{-1}\big(-\hat{\boldsymbol{\omega}}_{\boldsymbol{B}}\mathcal{I}\boldsymbol{\omega_B} + u_\tau\big), \\ {}^W\dot{R}_B &= {}^W R_B \hat{\boldsymbol{\omega}}_{\boldsymbol{B}}, \end{aligned} \tag{6}$$

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, $\mathbf{z_B} = {}^W R_B \mathbf{z_W}$ is the z axis in the body frame, $\mathcal{I}$ is the moment of inertia and $\hat{\boldsymbol{\omega}}_{\boldsymbol{B}}$ is the skew matrix obtained from $\boldsymbol{\omega_B}$, such that for any $a, b \in \mathbb{R}^3, \hat{a}b = a \times b$, where $\times$ denotes the cross product.

The controlller design presented in this section was based on [6], where a trajectory tracking controller is implemented for a differentially flat system such as a quadrotor. This controller is composed by two main loops, as represented in Figure 3: An inner loop for attitude stabilization that regulates rotation and angular velocity components, and an outer loop for position tracking that controls the position and velocity. At each time step, new inputs $\mathbf{u}$ are computed from the previous state $\mathbf{x}$, the reference trajectory position $\mathbf{p_{ref}}$ and the derivatives $\mathbf{v_{ref}}$, $\mathbf{a_{ref}}$ and $\dot{\mathbf{a}}_{\mathbf{ref}}$, as well as and the reference yaw ψ_{ref} and its derivative $\dot{\psi}_{ref}$.

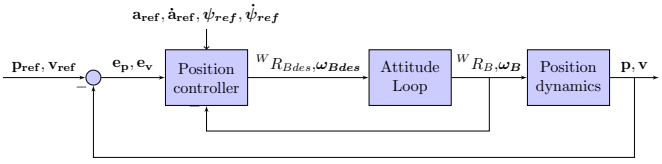


Figure 3: Representation of the inner and outer loops of the controller

Generally a controller regulates the behaviour of a system by forcing its input variables to zero. Therefore, in order to follow certain trajectories, we must provide the errors between the references and the actual values of the variables as inputs of the controller, so that when these go to zero, we know the system is following the references. To do so, we can compute the errors of position and velocity as

$$\mathbf{e_p} = \mathbf{p} - \mathbf{p_{ref}}, \qquad \mathbf{e_v} = \mathbf{v} - \mathbf{v_{ref}}. \tag{7}$$

From these, we can obtain the desired force vector as

$$\mathbf{F_{des}} = -K_p \mathbf{e_p} - K_v \mathbf{e_v} + mg\mathbf{z_W} + m\mathbf{a_{ref}}, \tag{8}$$

where K_p and K_v are positive definite matrices known as the position and velocity controller gains. The controller input u_1 is then defined as the projection of $\mathbf{F_{des}}$ onto $\mathbf{z_B}$ and can be written as $u_1 = \mathbf{F_{des}} \cdot \mathbf{z_B}$.

The other three inputs are related to the rotation errors, that depend on the desired Rotation matrix, ${}^W R_{Bdes}$, and angular velocity, $\boldsymbol{\omega_{Bdes}}$, which consequently can be determined from the additional references $\dot{\mathbf{a}}_{\mathbf{ref}}$, ψ_{ref}, and $\dot{\psi}_{ref}$. A detailed derivation of these expressions can be found in the thesis. The errors are then computed as

$$\mathbf{e_R} = \frac{1}{2}({}^W R_{Bdes}^T {}^W R_B - {}^W R_B^T {}^W R_{Bdes})^\vee, \tag{9}$$

where $^\vee$ represents the *vee map*, which is the inverse operation of the skew matrix, and

$$\mathbf{e}_\omega = \boldsymbol{\omega_B} - \boldsymbol{\omega_{Bdes}}. \tag{10}$$

In the end, we can determine the other control inputs as

$$\mathbf{u}_\tau = -K_R \mathbf{e_R} - K_\omega \mathbf{e}_\omega, \tag{11}$$

where K_R and K_ω are diagonal gain matrices.

Simulation tests are presented in the thesis with different trajectories generated in different optimization problems, where it can be seen that the system dynamic response is able to follow a reference without overshoot through a smooth and feasible trajectory, attenuating possible discontinuities in the reference.

4. 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 nonlinear optimization problem. The method described in this section is based on the work presented in [1], with the necessary changes to the problem in terms of constraints and problem structure.

4.1. 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, with $i = 1, ..., Q$, in the direction axis $j \in \{x, y, z\}$ and in the time step $k \in [1, \dots, K]$, and h is the discretization time. If we simplify the system dynamics in (6) so that $\dot{\mathbf{p}} = \mathbf{v}$ and $\dot{\mathbf{v}} = \mathbf{a}$, and assume that the acceleration is constant in between discretization time steps, i.e. $\mathbf{a}(t) = \mathbf{a}[k], \forall t \in [t_k, t_k + h]$, with $k = 0, ..., K-1$, then we may have the following discrete system dynamics:

$$\begin{aligned} v_{ij}[k+1] &= v_{ij}[k] + h a_{ij}[k], \\ p_{ij}[k+1] &= p_{ij}[k] + h v_{ij}[k] + \frac{h^2}{2} a_{ij}[k]. \end{aligned} \tag{12}$$

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

$$\begin{aligned} \boldsymbol{\chi} = &[a_{1x}[1], \dots, a_{1x}[K], a_{1y}[1], \dots, a_{1y}[K], a_{1z}[1], \dots, a_{1z}[K], \dots, \\ &\dots, a_{Qx}[1], \dots, a_{Qx}[K], a_{Qy}[1], \dots, a_{Qy}[K], a_{Qz}[1], \dots, a_{Qz}[K]]^T. \end{aligned} \tag{13}$$

In order to obtain expressions for the position and velocity that only depend on $\boldsymbol{\chi}$, the equations (12) can be modified as

$$\begin{aligned} v_{ij}[k] &= v_{ij}[1] + h(a_{ij}[1] + a_{ij}[2] + \dots + a_{ij}[k-1]), \\ p_{ij}[k] &= p_{ij}[1] + h(k-1)v_{ij}[1] + \frac{h^2}{2}((2k-3)a_{ij}[1] \\ &\quad + (2k-5)a_{ij}[2] + \dots a_{ij}[k-1]). \end{aligned} \tag{14}$$

If we repeat this 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

$$\begin{aligned} \mathbf{v} &= H_{11}\boldsymbol{\chi} + \mathbf{v_0}, \\ \mathbf{p} &= \frac{h^2}{2} H_{22}\boldsymbol{\chi} + hH_{33} \odot \mathbf{v_0} + \mathbf{p_0}, \end{aligned} \tag{15}$$

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.

4.2. Objective function

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 $\mathbf{F}$ is the total thrust, then the objective function can be formulated as

$$f_0 = \sum_{i=1}^{Q} \sum_{k=1}^{K} \|\mathbf{a_i}[k] + g\mathbf{z_W}\|^2, \tag{16}$$

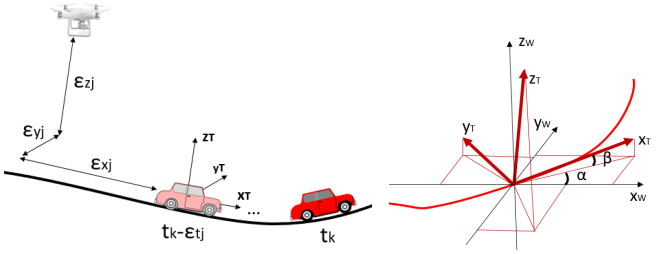
with $\mathbf{a_i}[k] = [a_{ix}[k], a_{iy}[k], a_{iz}[k]]^T$, which can be rewritten in the quadratic form $f_0(\boldsymbol{\chi}) = \boldsymbol{\chi}^T P \boldsymbol{\chi} + q^T \boldsymbol{\chi} + r$.

4.3. Constraints

The linear equality constraints are expressed as $A_{eq}\boldsymbol{\chi} = \mathbf{b_{eq}}$ and include the following:

1. Initial and final states - to define the initial and final positions, velocities and accelerations of each vehicle. Suppose that $k_I = 1, ..., K_I$ 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:

$$\begin{aligned} a_{ij}[k_I] &= a_{ij0}, & a_{ij}[k_F] &= a_{ijK}. \\ v_{ij}[k_I] &= v_{ij0}, & v_{ij}[k_F] &= v_{ijK}, \\ p_{ij}[k_I] &= p_{ij0}, & p_{ij}[k_F] &= p_{ijK}. \end{aligned} \tag{17}$$



(a) Position j in formation flight from ϵ_{tj}, ϵ_{xj}, ϵ_{yj} and ϵ_{zj}.

(b) Target frame.

Figure 4: Formation flight definition

2. Formation flight for target following - achieved by assigning specific parameters (ϵ_{tj}, ϵ_{xj}, ϵ_{yj}, and ϵ_{zj}, with $j = 1, ..., Q$) to different positions, as represented in Figure 4(a), so that the vehicles are distanced in time and space in relation to the target's position and orientation. Considering an additional coordinate system: the target frame, $\mathcal{T}$, as represented in Figure 4(b), the Rotation matrix denoted by ${}^W R_T[k] = [\mathbf{x_T}, \mathbf{y_T}, \mathbf{z_T}]$, that describes the orientation of the target in the world frame in a time step $k = 1, ..., K$, can be determined as:

$$\begin{aligned} \mathbf{x_T} &= \frac{\mathbf{p_T}[k] - \mathbf{p_T}[k-1]}{\|\mathbf{p_T}[k] - \mathbf{p_T}[k-1]\|}, \\ \alpha &= \arctan \frac{x_{Ty}}{x_{Tx}}, \\ \beta &= \arcsin x_{Tz}, \\ \mathbf{y_T} &= [-\sin(\alpha)\cos(\beta), \cos(\alpha)\cos(\beta), -\sin(\beta)]^T, \\ \mathbf{z_T} &= [\sin(\alpha)\sin(\beta), -\cos(\alpha)\sin(\beta), -\cos(\beta)]^T. \end{aligned} \tag{18}$$

where $\mathbf{p_T}$ is the position of the target and k and $k-1$ are consecutive time steps in $[1, ..., K]$. A general position j in formation flight described in the world frame is given by

$$\mathbf{p_{form_j}}[k] = \mathbf{p_T}\left[k - \frac{\epsilon_{tj}}{h}\right] + {}^W R_T\left[k - \frac{\epsilon_{tj}}{h}\right] \boldsymbol{\epsilon_{pj}}. \tag{19}$$

where $\boldsymbol{\epsilon_{pj}} = [\epsilon_{xj}, \epsilon_{yj}, \epsilon_{zj}]^T$ is a vector with the formation positions.

To assure that a vehicle i, with $i = 1, ..., Q$, follows the target in position j in formation flight, the following constraint must be implemented:

$$\mathbf{p_i}[k] = \mathbf{p_{form_j}}[k]. \tag{20}$$

Then, the linear inequalities are written as $A_{in}\boldsymbol{\chi} \leq \mathbf{b_{in}}$, where $\leq$ represents an element-wise inequality, and include the following:

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 $k = 1, ..., K$, we have the following constraints:

$$\mathbf{p_{min}} \leq \mathbf{p_i}[k] \leq \mathbf{p_{max}}, \tag{21}$$

$$\mathbf{v_{min}} \leq \mathbf{v_i}[k] \leq \mathbf{v_{max}}, \tag{22}$$

$$\mathbf{a_{min}} \leq \mathbf{a_i}[k] \leq \mathbf{a_{max}}, \tag{23}$$

$$\mathbf{j_{min}} \leq \mathbf{j_i}[k] \leq \mathbf{j_{max}}, \tag{24}$$

where $\mathbf{j}$ represents the jerk, which measures the variation of acceleration and can be computed as

$$\mathbf{j}[k] = \frac{\mathbf{a}[k] - \mathbf{a}[k-1]}{h}. \tag{25}$$

2. Minimum height during flight (H_{min}) - For a vehicle i, with $i = 1, ..., Q$, and in a time step k, this constraint can be written as

$$p_{iz}[k] \geq H_{min}. \tag{26}$$

Finally, the nonlinear constraints are defined as nonlinear functions of the form $c(\boldsymbol{\chi}) \leq 0$. The following constraints were 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 $[1, ..., K]$, this constraint can be formulated as

$$\|\mathbf{p_i}[k] - \mathbf{p_j}[k]\| \geq d_{saf}. \tag{27}$$

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 flight. Considering the situation where drone q cannot be seen in 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 150° ($\alpha_{cam} = 75°$ relative to $\mathbf{p_{Ti}}$). 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

$$\mathbf{p_{Ti}}[k]^T \mathbf{p_{qi}}[k] - \|\mathbf{p_{Ti}}[k]\| \|\mathbf{p_{qi}}[k]\| \cos(\alpha_{cam}) \leq 0. \tag{28}$$

This constraint is illustrated in Figure 5 where the camera's field of view is represented as a forbidden zone for any other drone.

4.4. Sequential Convex programming

Working directly with nonlinear constraints can be inefficient as they increase the complexity of the problem. As an alternative, we present an algorithm that replaces these constraints with linear approximations, where at iteration $s+1$, nonlinear constraints are linearised around the previous solution $\boldsymbol{\chi}^s$ using a first order Taylor expansion.

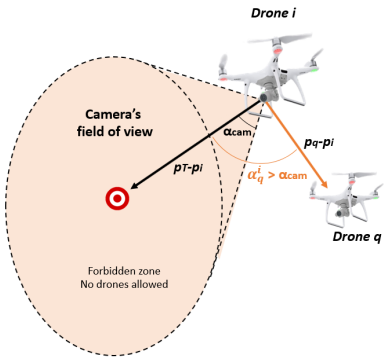


Figure 5: Illustration of the clear visibility constraint.

To linearise the collision avoidance constraint, we assume that $\mathbf{x}^s = \mathbf{p_i}^s[k] - \mathbf{p_j}^s[k]$ and $f_1(\mathbf{x}) = \|\mathbf{x}\|$, with $\mathbf{x} = \mathbf{p_i}[k] - \mathbf{p_j}[k]$. The linearised function $\tilde{f}_1(\mathbf{x})$ is given by

$$\tilde{f}_1(\mathbf{x}) = \|\mathbf{x}^s\| + \frac{\mathbf{x}^s}{\|\mathbf{x}^s\|}^T (\mathbf{x} - \mathbf{x}^s), \tag{29}$$

which leads to the following constraint

$$\tilde{f}_1(\mathbf{x}) \geq d_{saf}. \tag{30}$$

As for the clear visibility constraint, we consider that $\mathbf{x}^s = \mathbf{p_T}[k] - \mathbf{p_i}^s[k]$, $\mathbf{y}^s = \mathbf{p_q}^s[k] - \mathbf{p_i}^s[k]$ and $f_2(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y} - \|\mathbf{x}\|\|\mathbf{y}\| \cos(\alpha_{cam})$, with $\mathbf{x} = \mathbf{p_T}[k] - \mathbf{p_i}[k]$ and $\mathbf{y} = \mathbf{p_q}[k] - \mathbf{p_i}[k]$. The linearised function $\tilde{f}_2(\mathbf{x}, \mathbf{y})$ is then given by

$$\begin{aligned}\tilde{f}_2(\mathbf{x}, \mathbf{y}) &= \mathbf{x}^{sT}\mathbf{y}^s - \|\mathbf{x}^s\|\|\mathbf{y}^s\| \cos(\alpha_{cam}) \\ &+ \left(\mathbf{y}^s - \|\mathbf{y}^s\| \cos(\alpha_{cam}) \frac{\mathbf{x}^s}{\|\mathbf{x}^s\|}\right)^T (\mathbf{x} - \mathbf{x}^s) \\ &+ \left(\mathbf{x}^s - \|\mathbf{x}^s\| \cos(\alpha_{cam}) \frac{\mathbf{y}^s}{\|\mathbf{y}^s\|}\right)^T (\mathbf{y} - \mathbf{y}^s),\end{aligned} \tag{31}$$

from which we obtain the following constraint

$$\tilde{f}_2(\mathbf{x}, \mathbf{y}) \leq 0. \tag{32}$$

With these approximations, we can formulate the problem as a quadratic optimization problem, with a quadratic objective function and affine constraints. The SCP algorithm is described as follows:

1. Choose a starting point, which can be a solution without the nonlinear constraints.
2. From the previous point, compute the new A_{in} and $\mathbf{b_{in}}$ by adding the approximated constraints in (30) and (32).
3. Solve the quadratic optimization problem and obtain a solution $\boldsymbol{\chi}^s$.
4. Check if the stopping condition is satisfied, which only happens if:
   - $\boldsymbol{\chi}^s$ fulfils the nonlinear constraints.
   - convergence of the objective value is achieved, which occurs when $|f_0(\boldsymbol{\chi}^{s-1}) - f_0(\boldsymbol{\chi}^s)| < \delta$, with δ being a threshold parameter.

   If this is not satisfied, go back to step **2** and use the computed solution to linearise the next iteration.

4.5. Analysis of the optimization problem

We should aim to formulate the optimization problem as a convex program, as it guarantees that if we can find a local optimal solution, then this solution is also unique and global. Furthermore, convex problems are usually less complex and simpler to prove convergence. For an optimization problem to be identified as convex, the objective function should be convex, the equality constraints linear and the inequalities defined by convex sets.

In our problem, the objective function is convex if $\nabla^2 f(x) \succeq 0$, which only happens if $P \succeq 0$, i.e. P is a symmetric semi-positive definite matrix, which is indeed the case. 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. The nonlinearity can be avoided by applying the SCP algorithm. In this way, 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.

5. Online planning and control

In this Section, we address the same problem with a real-time trajectory planning and control algorithm by implementing a Nonlinear Model Predictive Controller that is able to simultaneously generate trajectories for a short time horizon and control the system in real time. The method implemented was based on the work presented in [15], adapted to the problem at hand as explained below.

The optimization problem is similar to the one implemented in Section 4, but with the difference that, in this case, we are explicitly considering the orientation of the vehicle and the position and orientation of the camera. Additionally, we are also including perception objectives that are related to how the camera of the UAV is looking at the target. For this purpose, it was necessary to consider the three coordinate systems represented in Figure 2: world frame $\mathcal{W}$, body frame $\mathcal{B}$, and camera frame $\mathcal{C}$.

5.1. System dynamics

In order for the algorithm to run in real time, it is important that the computation time at each time step is smaller than the sampling time, T_s, which can be achieved by reducing the number of state variables and simplifying the system dynamics. For this reason, we modified the system dynamics described in (6) by:

- using quaternions $\mathbf{q} = q_w + q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k}$ instead of Rotation matrices, which can also be represented by a vector $\mathbf{q}$ in the 3-Sphere $\mathbb{S}^3 = \{\boldsymbol{q} \in \mathbb{R}^4 : \|\boldsymbol{q}\| = 1\}$, with the quaternion multiplication operator denoted as $\otimes$.

- considering as control variables the mass-normalized thrust $\tilde{u}_1 = \frac{F_1+F_2+F_3+F_4}{m}$, where F_i is the thrust generated by the i^{th} motor and m is the mass, and the angular velocities $\boldsymbol{\omega_B} = [p, q, r]^T$ expressed in the body frame, instead of net thrust and body moments.

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

$$\mathbf{x} = [\mathbf{p_1}, ..., \mathbf{p_Q}, \mathbf{v_1}, ..., \mathbf{v_Q}, {}^W\mathbf{q}_{B1}, ..., {}^W\mathbf{q}_{BQ}]^T, \tag{33}$$

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 $i = 1, .., Q$, and the vector of control inputs is

$$\mathbf{u} = [\tilde{u}_{11}, ..., \tilde{u}_{1Q}, \boldsymbol{\omega_{B1}}, ..., \boldsymbol{\omega_{BQ}}]^T. \tag{34}$$

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

$$\begin{aligned} \dot{\mathbf{p}}_\mathbf{i} &= \mathbf{v_i}, \\ \dot{\mathbf{v}}_\mathbf{i} &= -g\mathbf{z_W} + R_q({}^W\mathbf{q}_{Bi})\tilde{u}_1\mathbf{z_W}, \\ {}^W\dot{\mathbf{q}}_{iB} &= \frac{1}{2}\Lambda(\boldsymbol{\omega_{Bi}})\,{}^W\mathbf{q}_{Bi}, \end{aligned} \tag{35}$$

where $R_q({}^W\mathbf{q}_{Bi})$ is the quaternion rotation operation and $\Lambda(\boldsymbol{\omega_{Bi}})$ is the skew-symmetric matrix in 4D of $\boldsymbol{\omega_{Bi}}$ given by

$$\Lambda(\boldsymbol{\omega_{Bi}}) = \begin{bmatrix} 0 & -p_i & -q_i & -r_i \\ p_i & 0 & r_i & -q_i \\ q_i & -r_i & 0 & p_i \\ r_i & q_i & -p_i & 0 \end{bmatrix}. \tag{36}$$

5.2. Perception Objectives

Additional perception objectives are included in the optimization problem, that are related to the deviation (and its derivative) of the center of the camera to the target, in order to guarantee that the UAV is looking at the target during formation flight and without oscillations. Note that we are assuming that the position and orientation of the camera w.r.t. the body is fixed throughout the flight.

Suppose that $\mathbf{p_i}$, $\mathbf{v_i}$ and ${}^W\mathbf{q}_{iB}$ are the position, velocity and orientation of drone i's body in the world frame, ${}^{\mathbf{Bi}}\mathbf{p_{Ci}}$ and ${}^{Bi}\mathbf{q}_{Ci}$ are the position and orientation of its camera w.r.t. the body frame and $\mathbf{p_T}$ and $\mathbf{v_T}$ are the position and velocity of the target in the world frame. Then, the position of the target in the camera frame is

$${}^{\mathbf{Ci}}\mathbf{p_T} = R_q({}^W\mathbf{q}_{Bi} \otimes {}^{Bi}\mathbf{q}_{Ci})^{-1}\left[\mathbf{p_T} - \left(R_q({}^W\mathbf{q}_{Bi})\,{}^{\mathbf{Bi}}\mathbf{p_{Ci}} + \mathbf{p_i}\right)\right]. \tag{37}$$

The projection of ${}^{\mathbf{Ci}}\mathbf{p_T}$ in the image plane coordinates (s_x, s_y) can be computed according to the classical pinhole camera model. Considering the focal lengths f_x and f_y for pixel rows and columns, these can be expressed as

$$s_x = f_x\frac{{}^{Ci}p_{Tx}}{{}^{Ci}p_{Tz}}, \qquad s_y = f_y\frac{{}^{Ci}p_{Ty}}{{}^{Ci}p_{Tz}}. \tag{38}$$

To further reduce the offset and oscillations of the camera view in respect to the target, we can also minimize the velocity of the projection onto the image plane, which is given by

$$\begin{aligned} \dot{s}_x &= f_x\frac{{}^{Ci}\dot{p}_{Tx}\,{}^{Ci}p_{Tz} - {}^{Ci}p_{Tx}\,{}^{Ci}\dot{p}_{Tz}}{{}^{Ci}p_{Tz}^2}, \\ \dot{s}_y &= f_y\frac{{}^{Ci}\dot{p}_{Ty}\,{}^{Ci}p_{Tz} - {}^{Ci}p_{Ty}\,{}^{Ci}\dot{p}_{Tz}}{{}^{Ci}p_{Tz}^2}, \end{aligned} \tag{39}$$

where ${}^{\mathbf{Ci}}\dot{\mathbf{p}}_{\mathbf{T}}$ is the derivative of (37) and can be computed as

$$
\begin{aligned}
{}^{\mathbf{Ci}}\dot{\mathbf{p}}_{\mathbf{T}} = R_q\Big(-\frac{1}{2}\Lambda(R_q({}^{Bi}\mathbf{q}_{Ci}^{-1})\boldsymbol{\omega_{Bi}})^{Bi}\mathbf{q}_{Ci}\Big){}^{\mathbf{Ci}}\mathbf{p}_{\mathbf{T}} + \\
R_q\Big({}^{W}\mathbf{q}_{Bi}\otimes{}^{Bi}\mathbf{q}_{Ci}\Big)^{-1}\Big[\mathbf{v_T} - \Big(R_q(\frac{1}{2}\Lambda(\boldsymbol{\omega_{Bi}})^{W}\mathbf{q}_{Bi})^{\mathbf{Bi}}\mathbf{p_{Ci}} + \mathbf{v_i}\Big)\Big]
\end{aligned}
\tag{40}
$$

Finally, we can obtain the perception parameters

$$
\mathbf{z_i} = [s_x, s_y, \dot{s}_x, \dot{s}_y]^T \tag{41}
$$

for each drone $i = 1, ..., Q$, which will be added in the objective function of the MPC algorithm.

5.3. NMPC algorithm

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

$$
\begin{aligned}
\min_{\mathbf{U}}\ & V_N(\mathbf{X},\mathbf{U},\mathbf{Z}), \\
\text{subject to } & \mathbf{r}(\mathbf{X},\mathbf{U},\mathbf{Z}) = 0, \\
& \mathbf{h}(\mathbf{X},\mathbf{U},\mathbf{Z}) \leq 0,
\end{aligned}
\tag{42}
$$

where $\mathbf{X} = [\mathbf{x_0}, ..., \mathbf{x_N}]$, $\mathbf{U} = [\mathbf{u_0}, ..., \mathbf{u_{N-1}}]$, and $\mathbf{Z} = [\mathbf{z_0}, ..., \mathbf{z_{N-1}}]$ are state, input and perception sequences for the time horizon with the components in (33), (34) and (41) respectively, $V_N(\mathbf{X},\mathbf{U},\mathbf{Z})$ is the objective function and $\mathbf{r}(\mathbf{X},\mathbf{U},\mathbf{Z})$ and $\mathbf{h}(\mathbf{X},\mathbf{U},\mathbf{Z})$ represent equality and inequality constraints that the system must obey. To solve this optimization problem, it is necessary to discretize the system dynamics 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 sequence of $N+1$ state predictions $\mathbf{X}$, with $N = T_h/T_s$. The components of the algorithm are described in the next pages.

5.3.1 Objective function

The objective function is formulated as a Linear-Quadratic Regulator and can be written as

$$
\begin{aligned}
V_N =& (\mathbf{x_N} - \mathbf{x_N^{ref}})^T\mathcal{Q}_{xN}(\mathbf{x_N} - \mathbf{x_N^{ref}}) + \sum_{i=0}^{N-1}\Big((\mathbf{x_i} - \mathbf{x_i^{ref}})^T\mathcal{Q}_{xi}(\mathbf{x_i} - \mathbf{x_i^{ref}}) + \\
& (\mathbf{z_i} - \mathbf{z_i^{ref}})^T\mathcal{Q}_{zi}(\mathbf{z_i} - \mathbf{z_i^{ref}}) + (\mathbf{u_i} - \mathbf{u_i^{ref}})^T\mathcal{R}_i(\mathbf{u_i} - \mathbf{u_i^{ref}})\Big),
\end{aligned}
\tag{43}
$$

where:

- $\mathcal{Q}_{xi}$, $\mathcal{Q}_{zi}$ and $\mathcal{R}_i$, with $i = 0, ..., N-1$, are the time-varying state, perception and input cost matrices and $\mathcal{Q}_{xN}$ is the final state cost penalty. These matrices can be tuned during flight in order to adjust the contribution of the state, perception and inputs reference and to obtain a desirable response;
- $\mathbf{x_i^{ref}}$, with $i = 1, ..., N$, and $\mathbf{z_i^{ref}}$ and $\mathbf{u_i^{ref}}$, with $i = 0, ..., N-1$, are the reference of state, perception and inputs that may also vary throughout the time horizon.

The state references $\mathbf{x_i^{ref}}$ have the purpose of describing the drones manoeuvres so that they follow the flight plan in Figure 1, 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 are defined for each drone and serve as a baseline for the trajectories to be generated by the MPC. As for the perception references $\mathbf{z_i^{ref}}$, these should be zero in this particular case, so that the camera center is aligned with the target and the image is as static as possible. We can also assume that the reference values for the inputs are zero.

5.3.2 Constraints

We implemented the following constraints:

- System dynamics - already expressed in (35). These are discretized using multiple shooting as a transcript method and a Runge Kutta integration scheme, which are executed with *ACADO*'s [17] generation code functions. More details on the ACADO implementation are presented in the next section.
- Control bounds - for a vehicle i, with $i = 1, ..., Q$, these can be expressed as

$$
\begin{aligned}
-\tilde{u}_{1max} \leq \tilde{u}_{1i} \leq \tilde{u}_{1max}, \\
-\boldsymbol{\omega_{max}} \leq \boldsymbol{\omega_{Bi}} \leq \boldsymbol{\omega_{max}}.
\end{aligned}
\tag{44}
$$

- State bounds - can be written as

$$
\begin{aligned}
\mathbf{p_{min}} \leq \mathbf{p_i} \leq \mathbf{p_{max}}, \\
-\mathbf{v_{max}} \leq \mathbf{v_i} \leq \mathbf{v_{max}}
\end{aligned}
\tag{45}
$$

- Collision constraint - similar to (27).
- Clear visibility - Similar to (28) but instead of evaluating the angle between the target and other drone's position w.r.t. its own position, this constraint uses the angle w.r.t. the position of its camera. Considering that 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

$$
\mathbf{p_{C_iT}} = \mathbf{p_T} - \Big(R_q\big({}^{W}\mathbf{q}_{Bi}\big)^{\mathbf{Bi}}\mathbf{p_{Ci}} + \mathbf{p_i}\Big), \tag{46}
$$

and the vector between the position of drone q in the world frame, $\mathbf{p_q}$, and the position of drone i's camera is

$$
\mathbf{p_{C_iq}} = \mathbf{p_q} - \Big(R_q\big({}^{W}\mathbf{q}_{Bi}\big)^{\mathbf{Bi}}\mathbf{p_{Ci}} + \mathbf{p_i}\Big). \tag{47}
$$

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

$$
\mathbf{p_{C_iT}}^T\mathbf{p_{C_iq}} - \|\mathbf{p_{C_iT}}\|\|\mathbf{p_{C_iq}}\|\cos(\alpha_{cam}) \leq 0. \tag{48}
$$

5.4. *ACADO* implementation

To formulate the NMPC problem, we used the *ACADO* toolbox, which is a software environment and algorithm collection able to generate C/C++ code in order to solve automatic control and dynamic optimization problems,

such as an MPC problem [17]. The algorithm was 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 formulate the problem in *ACADO*, the following steps had to be executed:

- Provide a set of differential states, $\mathbf{x}$, and control inputs, $\mathbf{u}$, and for this particular problem, a set of online data variables, such as the position and velocity of the target that may vary for each iteration;
- Provide a dynamic model $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t)$, such as (35);
- Define MPC parameters, that include the time horizon and sampling time, the objective function (43) and a set of constraints (as described before).

To test the algorithm, one has to define for each iteration the state, perception and input references, the cost matrices $\mathcal{Q}_x$, $\mathcal{Q}_z$ and $\mathcal{R}$, the bounds for the constraints and online data 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 6 shows a block diagram representative of the MPC controlled system.

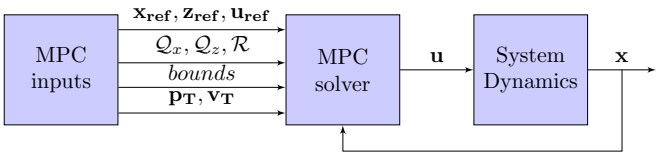


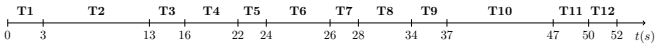
Figure 6: Diagram of the MPC controlled system

6. Simulation results

To validate the methods presented in Sections 4 and 5, these are tested in Matlab and using autopilot Software-In-The-Loop simulations. The latter are made in a Gazebo environment with the MULTIDRONE simulator [18], represented in Figure 7, where the UAVs are setup with a PX4 controller. All of the simulations presented in this Section were conducted on a Lenovo Laptop running Ubuntu 16.04 and equipped with an Intel Core i7-7700HQ CPU @2.80GHz and 16,00GB of RAM. Note that the SITL results should closely resemble the ones obtained with real experiments, as these employ the same algorithms and software for the autopilot and the physics engine faithfully reproduces the vehicles dynamics. We were able to record videos showcasing the tests presented in this Section, which are available at: https://fenix.tecnico.ulisboa.pt/homepage/ist178289/trajectory-planning-and-control-for-drone-replacement-during-formation-flight .

6.1. Tests with pre-planned trajectories

The optimization problem presented in Section 4 was implemented using Matlab's optimization toolbox and *Cplex* [19], where it was possible to generate optimal waypoint trajectories, that were then used as references for the PX4 controller in the SITL simulations. We will present a test made for a scenario with 3 drones following a target in a straight line, where drone 1 is replaced by drone 3. We planned a trajectory of 52 seconds with a fixed time step of $h = 0.5$ seconds, which was split into 12 intervals (according to Figure 1) with the following durations:



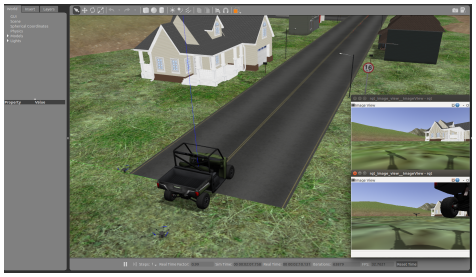


Figure 7: Gazebo environment with a MULTIDRONE simulator

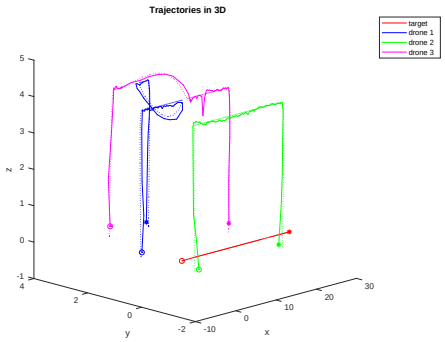
In the end, we obtained the simulation test showcased in the **Waypoint trajectories video**, where we can see that the drones were able to follow the flight plan previously described without any risk of collision and without being seen by the other drones in formation flight. This can also be verified in the plots from Figure 8, where small deviations are observed between the experimental and reference trajectories, which are more predominant during the flight transitions.

To check if the nonlinear constraints are satisfied, we plotted the distances between any two drones in the system (Figure 9) and saw that these are always above the minimum safety distance, thus ensuring collision avoidance. As for the clear visibility condition, to check if drone 3 is not detected during T_5 and T_6, we computed 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 during T_6 and T_7, we computed the angles between drone 1 and the target w.r.t drone 2 and 3. These are represented in Figure 10, 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 the video. Thus, we can say that the actual field of view angle is smaller than the one considered in the constraint.

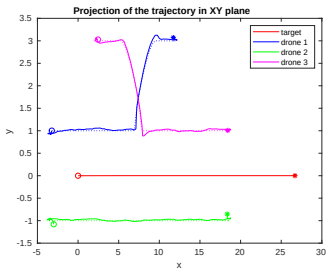
6.2. Tests with online planned trajectories

The MPC algorithm was also tested in Matlab and in SITL simulations. For the latter, we had to make some modifications, which included omitting the perception objectives, as the camera is no longer fixed to the body, and using the linear velocity predictions given by the MPC as control inputs for the PX4 controller instead of $\tilde{u}_1$ and $\boldsymbol{\omega_B}$ used in the Matlab simulation. We present a test made for a scenario with 2 drones following a similar flight plan as before with 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.

We obtained the simulation test showcased in the **MPC trajectories video**. From this, we can see that the proposed algorithm is both effective and implementable, as



(a) Trajectories in 3D



(b) Projection of the trajectories in the plane XY

Figure 8: Trajectories obtained with the pre-planned trajectories. Note that the *doted line* is the trajectory generated in Matlab, the *full line* is the trajectory obtained in the SITL simulation, the *circles* are the initial positions and the *stars* the final positions.

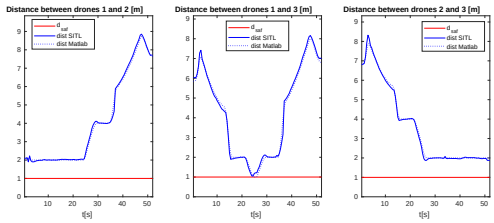


Figure 9: Distance between drones obtained with the pre-planned trajectory.

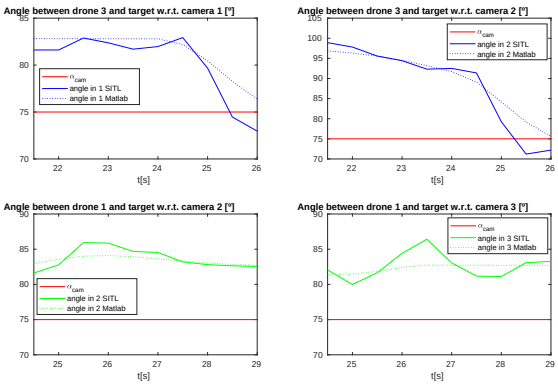
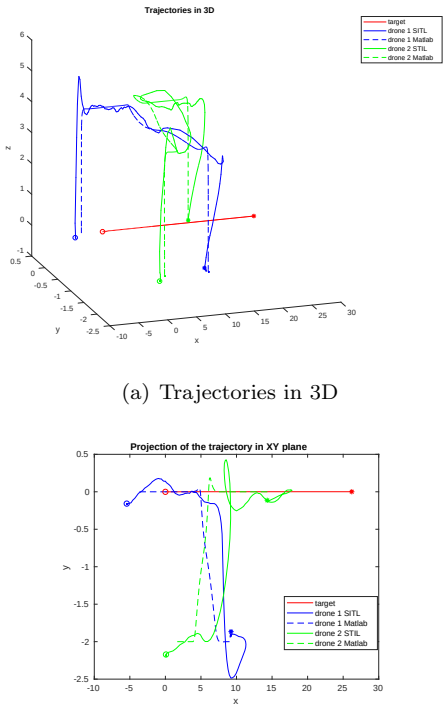


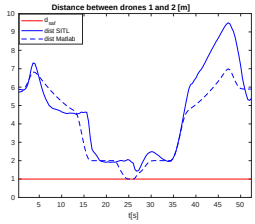
Figure 10: Clear visibility constraint obtained with the pre-planned trajectory.



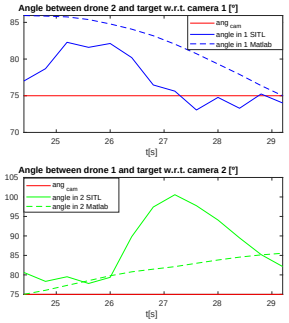
(a) Trajectories in 3D

(b) Projection of the trajectories in the plane XY

Figure 11: Trajectories obtained in the MPC simulation. Note that *circles* are the initial positions and the *stars* the final positions.



(a) Collision avoidance.



(b) Clear visibility.

Figure 12: Nonlinear constraints in the MPC simulations

feasible trajectories were computed in real time. The vehicles managed to execute the necessary manoeuvres and to exchange positions without colliding or being seen. These trajectories are plotted in Figure 11, where we can see that the STIL trajectories are not as smooth as the ones in Matlab, especially in the transition flight phases and they do not follow the MPC's initial reference as accurately.

Furthermore, we verified if the constraints for collision avoidance and clear visibility were satisfied by plotting the distances and the angles in Figure 12 and noted that while collision avoidance is verified in both simulations, clear visibility fails in the SITL simulation even though this is verified in the video. As for the computation time, this was around 0.06s in Matlab and 0.15s in the SITL results, which are lower than T_s $(0.4s)$. Therefore, we can conclude that the algorithm performs well, but could still be improved in order to increase its robustness.

7. 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. This was achieved in two ways: with a pre-planned trajectory generation method and with an online trajectory planning algorithm based on MPC. 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 the drones were able to execute the manoeuvres initially planned without any risk of collision and without being seen during the replacement process. We also proposed a tracking controller to follow the generated trajectories. In a second stage, we implemented an NMPC algorithm, taking into account the dynamic model of the system as well as an objective function and a set of constraints. We also tested this method in Matlab and in SITL simulations and concluded that the algorithm performs well, but is not robust as it may fail if the initial conditions are not adequate.

In terms of future work, there are still several lines of research that could be pursued, such as investigating alternative convex constraints to the nonlinear collision avoidance and clear visibility constraints, improving the robustness of our MPC algorithm and exploring decentralized MPC approaches. In terms of experimental results, further work could be dedicated to testing the situations presented in the SITL simulations in Section 6 with the generated waypoint trajectories and the MPC algorithm in real flight experiments with 2 and/or 3 drones and obtain experimental results.

References

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

[2] A. U. Raghunathan, V. Gopal, D. Subramanian, L. T. Biegler, and T. Samad. Dynamic Optimization Strategies for Three-Dimensional Conflict Resolution of Multiple Aircraft. *Journal of Guidance, Control, and Dynamics*, 2004.

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

[4] A. Richards and J. P. How. Aircraft Trajectory Planning With Collision Avoidance Using Mixed Integer Linear Programming. *American Control Conference (IEEE Cat. No.CH37301)*, pages 1936–1941 vol.3, 2002.

[5] T. Lee, M. Leoky, and N.H. McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). *49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425, 2010.

[6] D. Mellinger and V. Kumar. Minimum Snap Trajectory Generation and Control for Quadrotors. *IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011.

[7] P. Ru and K. Subbarao. Nonlinear Model Predictive Control Applied to Trajectory Tracking for Unmanned Aerial Vehicles. *AIAA Atmospheric Flight Mechanics Conference*, 2017.

[8] C. Sferrazza, M. Muehlebach, and R. D'Andrea. Trajectory Tracking and Iterative Learning on an Unmanned Aerial Vehicle using Parametrized Model Predictive Control. *56th IEEE Conference on Decision and Control (CDC)*, pages 5186–5192, 2017.

[9] M. Kamel, M. Burri, and R. Siegwart. Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles. *IFAC-PapersOnLine*, 50:3463–3469, 2017.

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

[11] Y. Kuwata, A. Richards, T. Schouwenaars, and J.P. How. Distributed robust receding horizon control for multivehicle guidance. *IEEE Transactions on Control Systems Technology*, 15(4):627–641, 2007.

[12] B. Penin, R. Spica, P. R. Giordano, and F. Chaumette. Vision-based minimum-time trajectory generation for a quadrotor UAV. pages 6199–6206, 2017.

[13] C. Potena, D. Nardi, and A. Pretto. Effective target aware visual navigation for UAVs. *European Conference on Mobile Robots*, 2017.

[14] M. Sheckells, G. Garimella, and M. Kobilarov. Optimal visual servoing for differentially flat underactuated systems. *IEEE International Conference on Intelligent Robots and Systems*, pages 5541–5548, 2016.

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

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

[17] D. Ariens, B. Houska, H. Ferreau, and F. Logist. ACADO Toolkit Users Manual, V 1.2.1beta, 2014. `http://www.acadotoolkit.org/`.

[18] Multidrone project. `https://multidrone.eu/`. Accessed: 2018-09-30.

[19] IBM Corp. IBM ILOG CPLEX Optimization Studio Getting Started with CPLEX for MATLAB V12.6, 2015.