Nonlinear Model Predictive Control for Formations of Multi-Rotor Micro Aerial Vehicles: An Experimental Approach

Erunsal, I. K.[1,2], Ventura, R.[2], Martinoli, A.[1]

[1] Distributed Intelligent Systems and Algorithms Laboratory, School of Architecture, Civil and Environmental Engineering, École Polytechnique Fédérale de Lausanne
Lausanne, Switzerland
`kagan.erunsal@epfl.ch, alcherio.martinoli@epfl.ch`

[2] Institute for Systems and Robotics, Instituto Superior Técnico
Lisbon, Portugal
`rodrigo.ventura@isr.tecnico.ulisboa.pt`

Abstract. In a world where the complexity and performance requirements of the tasks requested from micro aerial vehicles are continuously increasing, smooth design and deployment of multi-robot systems are gaining more significance. This paper tackles such challenging requirements by firmly adopting an architecture based on Nonlinear Model Predictive Control (NMPC). In order to efficiently design such architecture, we propose an approach emphasizing a closure of the reality gap between algorithmic design and physical experiments. More specifically, we use canonical system identification methods combined with additional calibration effort to enhance the faithfulness of our model in a high-fidelity simulation environment. By employing the accurate model obtained, we prototype our decentralized NMPC algorithm in a real-time iteration scheme. To improve further the performance, multi-modal, multi-rate, decentralized extended Kalman filters are integrated to the architecture. While experiments involving up to three quadrotors in high-fidelity simulation and reality outlined the approach's validity, they also pointed out its limitations when subtle effects generated by aerodynamic interactions among quadrotors are not taken into account in the control design.

Keywords: Formation control, multi-rotor micro aerial vehicles, nonlinear model predictive control, real-time iteration

1 Introduction

The last decade has been characterized by an increasing research effort on the control and coordination strategies for Micro Aerial Vehicles (MAVs) performing challenging missions such as scientific exploration, remote sensing, construction and search and rescue [1], [2]. Among the main design blocks of multi-MAV coordination strategies, formation control has a crucial role and it has been widely studied [3]. One of the promising methods to perform this task effectively leverages Nonlinear Model Predictive Control (NMPC), due to its architectural flexibility and ability to simultaneously satisfy the performance requirements and constraints of complex nonlinear systems [4]. Furthermore, if the number of robots in the system is large, the decentralized or distributed versions of NMPC (D-NMPC) are more suitable to subdivide the problem into decoupled pieces and reduce the interaction requirements [12]. In addition, in order to promote further autonomy, several researchers have investigated the formation control relying exclusively on relative localization and attempted to solve its limitations [15]. However, not only because of the implementation complexity but also the safety concerns related to multi-MAV systems, it is currently very time-consuming and challenging to iteratively prototype and validate any control and coordination strategy in reality. One of the techniques that makes such iterative prototyping framework more efficient is to bridge physical experiments and algorithmic design, typically carried out with mathematical formalism and highly abstracted simulations (e.g., bodiless agents implemented in Matlab) with an embodied, high-fidelity simulation environment. However, choosing an appropriate high-fidelity simulator is not enough: additional calibration efforts are typically needed to increase the faithfulness of the simulation. This paper studies the effectiveness of such framework in the design and validation of decentralized NMPC-based control architectures for a multi-MAV system. As a concrete case study for our approach, we have chosen a leader-follower formation control architecture due to its implementation simplicity and considered three MAVs in total. While the leader is responsible for steering the formation to a desired point, the followers' duty is to maintain the formation as depicted in Fig. 1. There are various successful implementations of MPC-based strategies for a single MAV performing trajectory tracking [5], robust trajectory following [6] and perception-aware motion generation [7]. Furthermore, there exist different successful validations of multi-robot formation control concerned with the formations in motion [8], encirclement [9], rendezvous [10], outdoor flocking [18] and formation maintenance [19]. However, although some of them introduce basic system identification techniques, none of them present a complete framework for closing the reality gap, including system identification, calibration of simulation and validation of control and estimation algorithms in reality. Additionally, none of them focuses on the intersection between D-NMPC and formation control of MAVs leveraging exclusively relative localization systems.

2 Technical Approach

Any MPC-based strategy requires a realistic model of the system dynamics to be controlled. To this purpose, we adopted the approach proposed in [5] given the fact that the hardware and software setup is very similar to ours. Note that, for our vehicle model, the same notation as in [11] is used for the vectors, coordinate frames and rotation matrices. However, there are a few main differences as well. First, the state $\boldsymbol{s}$ is defined as in Eq. (1):

$$\boldsymbol{s} = [{\boldsymbol{x}_{b/n}^{n}}^{T} \; {\boldsymbol{v}_{b/n}^{n}}^{T} \; {\boldsymbol{t}_{b/n}^{n}}^{T}]^{T} \tag{1}$$

where $\boldsymbol{x}_{b/n}^{n}$ is the absolute position, $\boldsymbol{v}_{b/n}^{n}$ is the velocity and $\boldsymbol{t}_{b/n}^{n}$ is the attitude of the quadrotor in the body-fixed frame $\{b\}$ with respect to the inertial frame

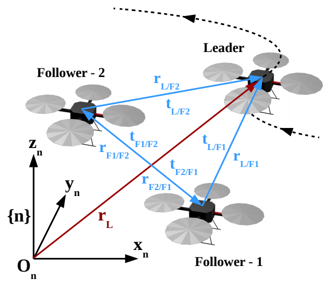


Fig. 1. Formation control problem with one leader and two followers with the inter-vehicle positions and trajectory followed

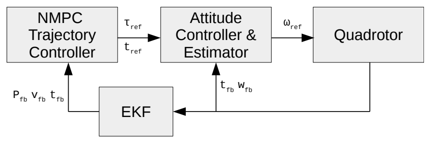

NMPC Trajectory Controller
τ_{ref}
t_{ref}
Attitude Controller & Estimator
ω_{ref}
Quadrotor
EKF
P_{fb} v_{fb} t_{fb}
t_{fb} w_{fb}

Fig. 2. Cascaded control/estimation architecture for the individual quadrotor

$\{n\}$ (lower index) expressed in the same frame $\{n\}$ (upper index). Second, the model now includes attitude dynamics as three first-order closed-loop system equations. The complete model, which is valid for the leader, is given in Eqs. (2)-(4).

$$\dot{\boldsymbol{x}}^n_{b/n} = \boldsymbol{v}^n_{b/n} \tag{2}$$

$$m_b \dot{\boldsymbol{v}}^n_{b/n} = m_b \boldsymbol{g} + \boldsymbol{R}^n_b \boldsymbol{F}^b_{b/n} \tag{3}$$

$$\dot{\boldsymbol{t}}^n_{b/n,i} = \frac{1}{\tau_i}(k_i \boldsymbol{t}_{ref,i} - \boldsymbol{t}^n_{b/n,i}) \quad \text{for} \quad i = 1,2,3 \tag{4}$$

where m_b is the mass, $\boldsymbol{g}$ is the gravitational acceleration, $\boldsymbol{R}^n_b$ is the rotation matrix, $\boldsymbol{F}^b_{b/n}$ is the applied body forces, τ_i's are the time constants and k_i's are the gains of the first order dynamical model for roll, pitch and yaw angles. In order to predict the system evolution in NMPC, the parameters to be identified are the mass m_b, gains k_i, time constants τ_i and the conversion parameters from required thrust to attitude commands (throttle) as a function of battery voltage. Furthermore, in order to obtain an accurate simulation of the closed-loop system, the following characteristics should also be acquired: the torque constant T_q and thrust constant T_k of the propellers, the inertia I_b of the quadrotor, the lumped Position-Derivative (PD) attitude controller's parameters K_p and K_d for each Euler angle to achieve the identified attitude response. The corresponding experiments to obtain these parameters will be explained in Section 3.

Once the vehicle model has been selected and calibrated, the optimization problem to be tackled with NMPC can be generated for the trajectory tracking quadrotors. The problem is formulated in Eq. (5), for additional details, refer to [12]. The controller structure is designed in a cascaded way as shown in Fig. 2, inspired by [5]. This allows us to separate a high(er) frequency task (attitude control) from a low(er) frequency one (trajectory control).

For this problem, the outputs of NMPC, $\boldsymbol{u}$, are the reference thrust, roll and pitch angles. In addition, the cost functions $\boldsymbol{c_k}$ and $\boldsymbol{c_N}$ minimize the trajectory errors, non-gravitational desired thrust, i.e. the thrust spent on maneuvers, roll and pitch angles and their switching rates based on the selected vehicle model f_k. The stage inequality constraints h_k include the input and state bounds corresponding to the actuation limits and comfort constraints. In addition, the terminal constraint g_N is selected as a state set by considering the stability analysis.

$$\begin{aligned}
\underset{\substack{\boldsymbol{x}(1),\ldots,\boldsymbol{x}(N)\\ \boldsymbol{u}(0),\ldots\boldsymbol{u}(N-1)}}{\text{minimize}} \quad & \sum_{k=0}^{N-1} c_k(\boldsymbol{x}(k),\boldsymbol{u}(k),\boldsymbol{r}(k)) + c_N(\boldsymbol{x}(N),\boldsymbol{u}(N),\boldsymbol{r}(N)) \\
\text{subject to} \quad & \boldsymbol{x}(k+1) = f_k(\boldsymbol{x}(k),\boldsymbol{u}(k)), \quad k = 0,..,N-1 \\
& h_k(\boldsymbol{x}(k),\boldsymbol{u}(k)) \leq 0, \quad k = 0,..,N-1 \\
& g_N(\boldsymbol{x}(N)) \leq 0
\end{aligned} \tag{5}$$

To conduct the experiments accurately, a multi-modal, multi-rate Extended Kalman Filter (EKF) has been designed to filter out the noise on the perception. This filter is multi-modal because different dynamics are valid while flying and landed; it is multi-rate because the sampling rates of the various sensors used for navigation are different. The parameters of this filter are tuned based on the relative magnitudes of the measured process and sensor noises levels.

Considering the designed setup, various trajectory tracking experiments are carried out by generating trajectories between successive way points. These experiments will be presented in detail in Section 3.

Once the vehicle model and simulation are calibrated, the prototyping framework is ready for the multi-robot experiments. As introduced in Section 1, the leader-follower formation of multiple quadrotors is controlled by fully decentralized NMPC with the help of multi-modal, multi-rate local EKF. Since the architecture does not involve any inter-vehicle communication, the role of EKF is to filter out the noise in the relative sensing measurements and to estimate the linear and yaw velocity of the neighbor vehicles accurately. Note that the controller of the followers has also a cascaded structure as that of the leader. However, the followers include not only their own system dynamics but also the constant-velocity dynamics of all their neighbors in the formation. This assumption is valid if the acceleration of the vehicles is limited to low values, which is the case, for instance, during indoor flights. Another important point is that all states are now written with respect to a stationary, inertial frame, which we call MPC inertial frame $\{m\}$, where we assume that the initial pose of each vehicle corresponds to the zero vector at the beginning of the motion. This frame also facilitates the transfer of variables from estimator to controller since the EKF needs an inertial frame. In addition, two more frames needed to be included in order to write the model and controlled variables completely. The first one is the MPC body frame $\{d\}$, which is anchored to the body throughout the MPC horizon, and the second is the MPC control frame $\{c\}$, whose orientation is roll- and pitch-free while still being anchored to the body frame $\{d\}$. The latter frame is considered due to the fact that roll and pitch angles of a quadrotor cannot be controlled directly when performing 3D formation control, consequently, the control variables (relative positions) are defined here. All frames are represented in Fig. 3. The described model for the followers is given in Eqs. (6)-(11).

$$\dot{\boldsymbol{v}}^m_{d/m} = \frac{\boldsymbol{R}^m_d}{m_b}\boldsymbol{F}^d_{d/m} + \boldsymbol{g} \tag{6}$$

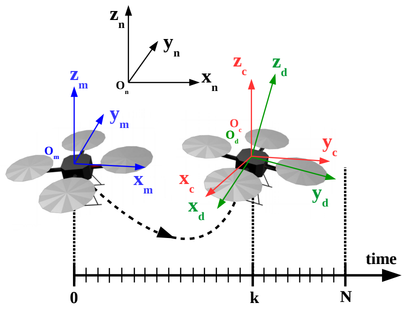


Fig. 3. MPC coordinate frames in a horizon of length N: $\{m\}$ (blue), $\{d\}$ (green), $\{c\}$ (red).

Fig. 4. Helipal Storm Drone-4 v3

$$\dot{\boldsymbol{t}}^m_{d/m,i} = \frac{1}{\tau_i}(k_i \boldsymbol{t}_{ref,i} - \boldsymbol{t}^m_{d/m,i}) \quad \text{for} \quad i = 1,2,3 \tag{7}$$

$$\dot{\Delta\boldsymbol{x}}^m_{d/m,j} = \boldsymbol{v}^m_{d/m,j} - \boldsymbol{v}^m_{d/m} \quad \text{for} \quad j = 1, ..., N_{nh} \tag{8}$$

$$\dot{\boldsymbol{v}}^m_{d/m,j} = \boldsymbol{0} \quad \text{for} \quad j = 1, ..., N_{nh} \tag{9}$$

$$\dot{\psi}_j = w_{z,j} \quad \text{for} \quad j = 1, ..., N_{nh} \tag{10}$$

$$\dot{w}_{z,j} = 0 \quad \text{for} \quad j = 1, ..., N_{nh} \tag{11}$$

where Δx, v_j, ψ_j, w_j and N_{nh} are the relative positions, absolute velocity, yaw angle and yaw rate of the neighbor vehicles expressed in $\{m\}$ and the number of neighbors in the formation, respectively. For the definitions of variables in Eqs. (6) and (7), please refer to Eqs. (3) and (4). Note that, for formation control, the cost function, has a similar structure as Eq. (5), as it minimizes the relative position and velocity errors, non-gravitational desired thrust, roll and pitch angles. Additionally, the inequality constraints include the state and input bounds according to the actuation and safety limits. Note that, the relative positions are included in the state vector and can therefore be bounded in order to obtain safe formations.

Here, it is worth mentioning the solution strategy used for NMPC. We chose a strategy based on Real-Time Iteration (RTI), as it is arguably the most successful and widely exploited approach to efficiently solve NMPC problems in real-time [20]. To apply this method, initially, a discretization is performed on the system model, constraints and cost function in order to obtain a structured Non-Linear Program (NLP). Next, the NLP is sequentially approximated by Quadratic Programs (QPs) using linearization techniques. After this step, a condensing method is applied to the QP to take advantage of fast condense QP solvers. Generally, a warm initialization is leveraged considering the fact that the optimal solution to the current iteration is very similar to the previous one. At this stage, a linear algebra solver exploits the QP to obtain a Newton direction for the solution. The resulting output is selected as an initial guess to the next QP and this procedure continues until the convergence is achieved. The optimality of the solution is measured by the Karush-Kuhn-Tucker (KKT) value.

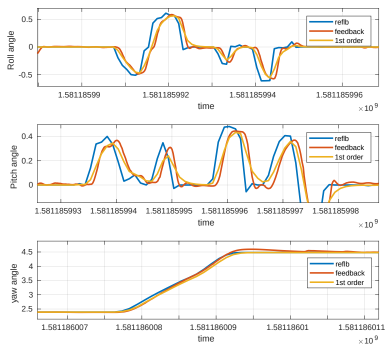


Fig. 5. Attitude subsystem identification results: reference, feedback and first order fitted model

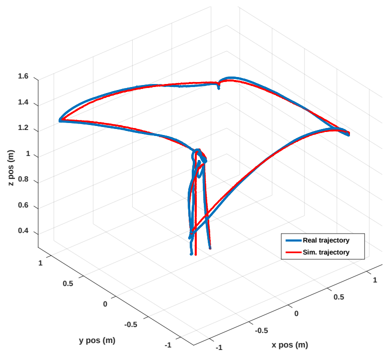


Fig. 6. Trajectory comparison between reality and simulation

The details of the RTI scheme which is repeated for each control iteration and tailored for this problem are explained in Algorithm 1.

Algorithm 1 RTI scheme

```
$X^g \leftarrow x^g$ {Initial guess of evolution of states}
$U^g \leftarrow u^g$ {Initial guess of evolution of inputs}
$X_0 \leftarrow x_0$ {Feedback obtained, initial condition}
$P_s \leftarrow p_s$ {Parameters and online data}
**while** $(t == 0 \ || \ t_{CPU} \leq t_s)$ && $KKT \geq threshold$ **do**
    {Initially iterate until obtain a low KKT result, for the next iterations check CPU time, control sample time and KKT to finish}
    $(X^*, U^*) \leftarrow SQP\_step(X^g, U^g, X_0, P_s)$ {Obtain optimal states and inputs by solving QP}
    **if** $(X^*, U^*)$ is infeasible || solver_error **then**
        $sflag \leftarrow true$ {Solver error flag}
        **break**
    **end if**
    $(X^g, U^g) \leftarrow (X^*, U^*)$ {Warm start for next sub-iteration}
**end while**
**if** $sflag == true$ **then**
    **Apply** $U^g(1)$ {Apply the solution found in previous iteration for this time step}
    $(X^g, U^g) \leftarrow X^g_s, U^g_s$ {Warm start for next iteration: time-shifted version of the previous guess}
**else**
    **Apply** $U^*(0)$ {Apply the first input of optimal solution}
    $(X^g, U^g) \leftarrow X^*_s, U^*_s$ {Warm start for next iteration: time-shifted version of the optimal solution}
**end if**
```

3 Experiments and Results

The quadrotors employed in the experiments are Helipal Storm Drone-4 v3 endowed with a Pixhawk Cube-2 autopilot, a Raspberry PI 3B onboard computer, an IMU, an optical flow sensor and a external digital compass as shown in Fig. 4. The 3D position information is generated by a Motion Capture System (MCS) and the optic-flow velocity is emulated by the same system due to its reliability. The employed MCS allows for millimeter accuracy in the measurement of the vehicles pose with an update rate of 60 Hz. All computations are carried out onboard by leveraging the Robot Operating System (ROS). As solver, the nonlinear OCP solver ACADO [21] is used with the active set QP solver qpOASES [22]. A control sample time of 0.05 seconds and a horizon length of 15 are chosen, resulting in a 0.75 seconds of total prediction horizon.

3.1 System identification

The propulsion subsystem of the drone (T_q, T_k and Thrust-Throttle-Voltage characteristics) is identified by using a RC Benchmark 1585 propeller setup [16]. Next, the mass is simply measured by a digital scale and the inertia is estimated by adopting a double-arm pendulum setup designed according to [13]. Furthermore, the attitude subsystem parameters are found based on the closed-loop position tests by fitting first-order responses to attitude references. Fig. 5 shows sample results of such system identification campaign. By leveraging this information, the lumped PD parameters of the autopilot are characterized and integrated into the simulation. Finally, all noise levels of sensors are measured in steady state and introduced into the simulation. The results of the parameter identification procedure are summarized in Table 1. Note that the full parameter

Table 1. Results of parameter identification

Parameter	Value	Unit
Mass, m	1.37	kg
Inertia, I_{xx}, I_{yy}, I_{zz}	0.0123, 0.0136, 0.0107	kgm^2
Thrust constant T_k	$9.803 * 10^{-8}$	N/rpm^2
Torque constant T_q	$1.507 * 10^{-9}$	Nm/rpm^2
Thrust-Throttle-Voltage characteristics	3rd order polynomial	-
First order time constants τ_i, i=1,2,3	0.1203, 0.1303, 0.0651	-
First order gains k_i, i=1,2,3	1.01, 1.02, 0.99	-
Lumped proportional gains $K_{p,i}$, i=1,2,3	3.0, 3.0, 0.5	-
Lumped derivative gains $K_{p,i}$, i=1,2,3	0.4, 0.4, 0.4	-

identification is carried out only for one quadrotor since the other ones employ exactly the same equipment (e.g., frame, motors, payload etc.). Only the mass is re-measured for the other drones. Although we are aware of the fact that there might be slight manufacturing variations among the vehicles, we observed that our NMPC architecture is not sensitive to them.

3.2 Calibration of simulation

With the fully identified vehicle model at hand, the trajectory tracking of the drone is achieved by generating six way points in a 5 m^3 volume and operating the decentralized NMPC-EKF architecture implemented in ROS. Data are collected and compared with the simulation performed in Webots [14], an open-source high-fidelity robotics simulator. By substituting the Euler angle biases with data measured on a real quadrotor and manually fine-tuning the thrust bias, good levels of trajectory matching between reality and simulation can be obtained, as illustrated in Fig. 6. The average error between the simulated and real closed-loop trajectories is about 0.06 m. In addition, the control inputs, i.e. thrust, roll and pitch references, for the real and simulated quadrotors are depicted in Fig. 7 for the sake of completeness. As can be observed, they are quite synchronous. The accompanying video further illustrates the faithfulness of the simulation.

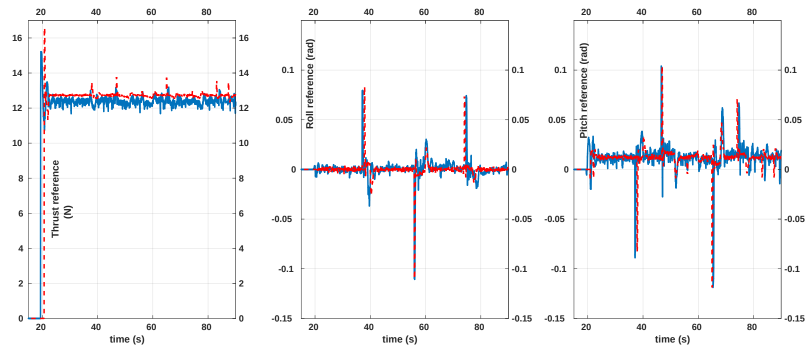
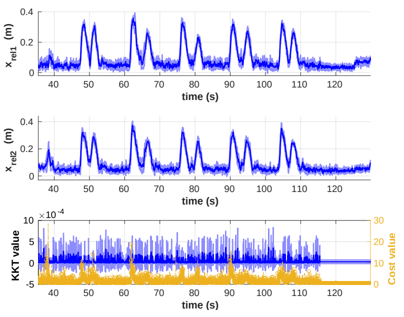


Fig. 7. Thrust, roll and pitch references sent to autopilot, generated by NMPC. Blue-solid: real data, red-dashed: simulated data



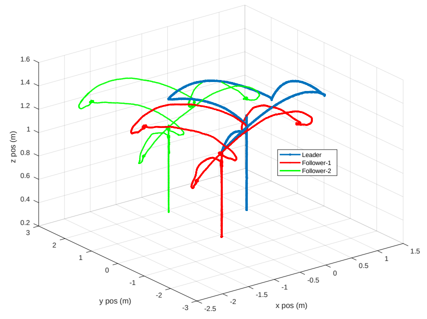


Fig. 8. Formation control results for multiple runs

Fig. 9. Leader and follower trajectories in simulation

3.3 Multi-robot formation control in simulation

We have carried out various formation control experiments in Webots, all involving one leader and two followers, as illustrated in Fig. 11. Note that in these simulations, 3D relative position and orientation information is generated by a 3D relative range and bearing sensor emulating the real system proposed in [17]. The trajectories in one experiment are shown in Fig. 9: the leader follows a prescribed trajectory and followers try to maintain a triangular formation by only relative sensing and estimation. The formation control errors, i.e. follower-leader and follower-follower, together with the solver's quality outputs, i.e. the cost and KKT values, over ten runs are shown in Fig. 8 for one of the followers. As can be seen, the average steady-state norm-error is around 0.05 m while the solver always converges. The quality of the formation control is also visualized in the enclosed video leveraging the calibrated high-fidelity simulation tool.

3.4 Multi-robot formation control in reality

Two drones, i.e. one leader and one follower, are employed in order to validate the concept. This setup is illustrated in Fig. 10. Two main experiments are designed and executed in order to clearly distinguish the effects of aerodynamic formation disturbance: formation control with a manually translated leader and with an autonomously flying leader. They will be elaborated in the following sections.

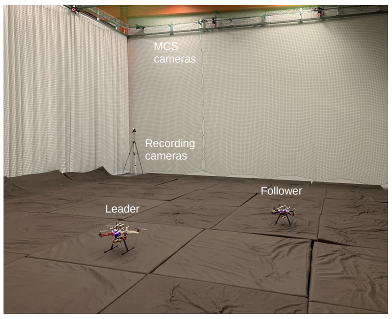


Fig. 10. Flight arena with one leader and one follower vehicles

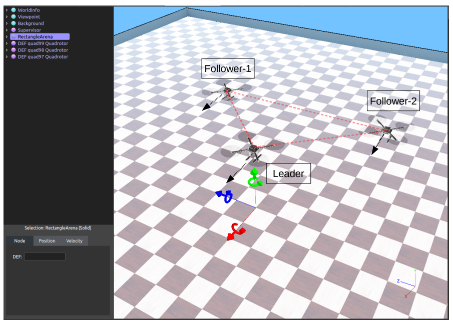


Fig. 11. Webots simulator with one leader and two follower vehicles

3.4.1 Formation control with manually translated leader

For this test, the leader is initially placed at a take-off height of 1 m. Since the follower's formation control algorithm is agnostic to absolute position sensing, the take-off is performed by a separate PID controller. Then, a switching algorithm accomplishes a smooth transition to the NMPC and the follower vehicle engages to the leader. The engagement relative position is selected as [2.5 -2.5 0] m. Next, the leader is manually and randomly moved in a confined space of 1.5 m^3. The tests are conducted three times and no significant differences among them are observed. The result of one of the trails is shown in Fig. 12. Note that the enclosed video also demonstrates this experiment. As can be observed

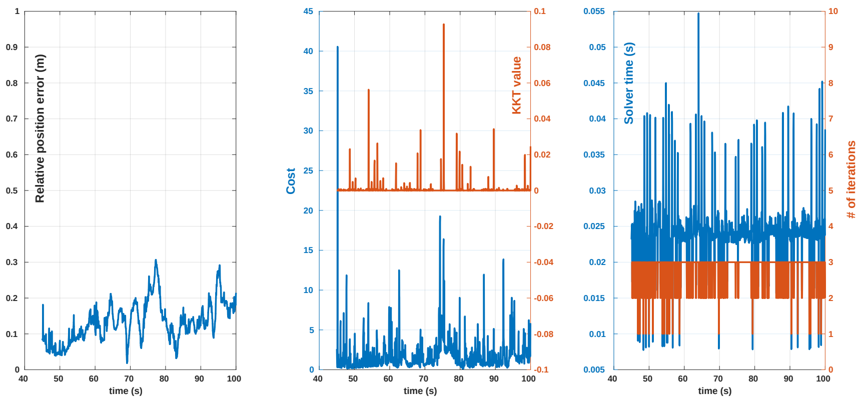


Fig. 12. Performance of formation with manually translated leader

from the left-most plot, the average error in 3D is about 0.14 m. In transition, this error can go up to 0.3 m; however, convergence is achieved when the leader stops and the average stationary error of 0.11 m is maintained. Concerning the solution quality, the KKT value is always lower than 0.1, and except occasional peaks, it stays smaller than 0.01. For real experiments with high dynamics such as that of a multi-MAV system, these values are acceptable. It is also clear that the cost function value converges towards zero after each perturbation. Lastly, thanks to the RTI algorithm, the solver time almost always remains lower than the control sampling period, which is 0.05 s. The overall results show that a satisfactory following performance is achieved in the absence of mutual aerodynamic disturbance.

3.4.2 Formation control with autonomously flying leader

The leader vehicle is controlled by the NMPC controller and follows a predefined 3D trajectory consisting of six way points in a limited space of 3.5 m^3. This trajectory is initialized after the follower is engaged to the leader with a relative position of [3.6 -2.8 0] m. Since the lateral directions of a quadrotor are highly sensitive to disturbances, the first observation is that the air flow generated by the neighbor drone creates a stochastic oscillatory effect in these directions. The results of one representative experiment are depicted in Fig. 13. This experiment is also included in the video enclosure. The main difference compared

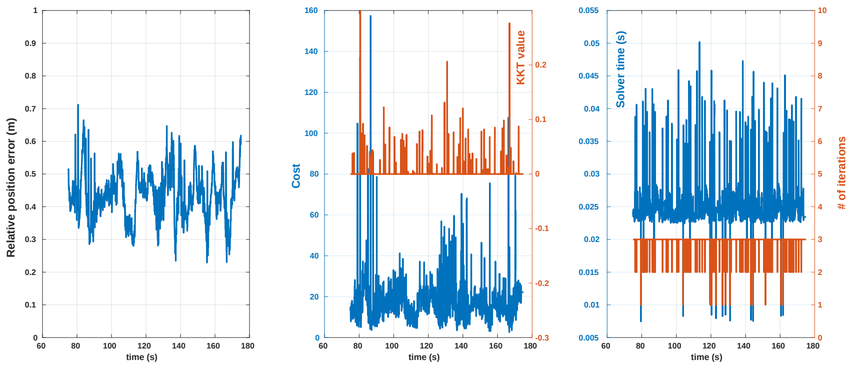


Fig. 13. Performance of formation with autonomously flying leader

to the previous experiment is that the relative position errors significantly rise, reaching an average error of 0.44 m (in spite of the fact that convergence is maintained). There are two main causes of this increase: first, the follower is vastly affected by the wind disturbance generated by the leader. These perturbations appear to be very turbulent and are not included in the model of the follower and leader. Second, due to the high gain control of the leader, i.e. in order to reject disturbances as much as possible, the leader does no longer operate in a low-acceleration regime, which leads to a further invalidation of the neighbor vehicle model in the NMPC computations. Finally, we can observe that both the average cost function and KKT values grow due to the highly nonlinear motion. Despite this, it is clear that the RTI scheme is able to keep the solver time under the control sample time, which is crucial for real-time control.

4 Conclusion and Outlook

Carrying out this work has allowed us to gain the following experimental insights. *Firstly*, during the real experimental validation on trajectory tracking of a single drone, it was clear that even an uncalibrated simulation performed in the high-fidelity simulator Webots coupled with ROS was already very beneficial from a software prototyping perspective (the very same code can be run in simulation and reality). Nonetheless, due to the model-based approach chosen, the differences between the trajectories generated in simulation and those of the real system were still significant. We managed to reduce the reality gap by following a two-stage calibration strategy: first, by identifying the physical parameters of the vehicle model underlying our NMPC strategy with canonical system identification algorithms for both real and simulated vehicles; second, by manually tuning simulation parameters in order to match the produced closed-loop trajectories. At the end of such two-stage process, it was straightforward to design, iteratively optimize, and validate control and estimation algorithms both in high-fidelity simulation and physical reality. *Secondly*, projecting these findings to multi-robot systems, the transition was found to be very smooth. In simulation, the followers are successful in tracking the leader, especially in smooth trajectory regions. However, due to the constant velocity assumption of the predicted trajectories in NMPC, one can see transient divergence on the sharp turns. This is an expected outcome and a drawback of a fully decentralized architecture which by design does not leverage communication. The impact of such choice is, however, insignificant for low acceleration flights, a valid assumption for some scenarios such as patrolling. On the positive side, since the computation and communication overhead of decentralized NMPC is considerably lower than that of distributed architectures, such choice shows good scalability towards systems involving larger number of vehicles. *Thirdly*, the experiments in reality carried out with both quadrotors flying outlined that the mutual aerodynamic disturbances significantly affect the reality gap, and in turn the formation performance, since they are not modeled in the controllers and they invalidate the constant velocity assumption. *Finally*, we believe that these results can be improved by integrating a robust framework to the system, for instance, a disturbance mapping/observer or a robust version of NMPC. This measure should reduce the model uncertainties and the impact of disturbances (causing a decrease in overshoots and steady-state errors) and in turn results in a diminution of formation errors. An additional performance improvement can be achieved through a distributed architecture by leveraging communication between vehicles. Not only the predicted trajectories but also the estimated disturbance information would be shared to enhance the model synchronization.

Acknowledgement

This work has been partially sponsored by the FCT grant [PD/BD/135151/2017], the FCT doctoral program RBCog and the FCT project [UID/EEA/50009/2013] [3].

References

1. Bemporad, A., Rocchi, C. (2011). Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles. 50th IEEE Conference on Decision and Control, 7488-7493.

[3] Additional information about this project can be found here: https://www.epfl.ch/labs/disal/research/quadrotorformationmpc/

2. Monteri, A., Freddi, A., Longhi, S. (2015). Nonlinear decentralized model predictive control for unmanned vehicles moving in formation. Information Technology and Control, 44(1), 89-97.
3. Chao, Z., Zhou, S. L., Ming, L., Zhang, W. G. (2012). UAV formation flight based on NMPC. Mathematical Problems in Engineering, 1-15.
4. Eren, U., Prach, A., Koer, B., Rakovi, S., Kayacan, E., Akmee, B.(2017). MPC in aerospace systems: Current state and opportunities. Journal of Guidance, Control and Dynamics, 40(7), 1541-1566.
5. Kamel, M., Burri, M., Siegwart, R. (2017). Linear vs nonlinear MPC for trajectory tracking applied to rotary wing MAVs. IFAC-PapersOnLine, 50(1), 3463-3469.
6. Kamel, M., Stastny, T., Alexis, K., Siegwart, R. (2017). Robot Operating Ststem(ROS): The Complete Reference, Vol.2, Model predictive control for trajectory tracking of UAV using ROS, Springer, Cham, 3-39.
7. Falanga, D., Foehn, P., Lu, P., Scaramuzza, D. (2018). PAMPC: Perception-aware model predictive control for quadrotors. IEEE/RSJ International Conference on Intelligent Robots and Systems, 5200-5207.
8. Van Parys, R., Pipeleers, G. (2017). Distributed MPC for multi-vehicle systems moving in formation. Robotics and Autonomous Systems, 97, 144-152.
9. Hafez, A. T., Marasco, A. J., Givigi, S. N., Iskandarani, M., Yousefi, S., Rabbath, C. A. (2015). Solving multi-UAV dynamic encirclement via MPC. IEEE Transactions on Control Systems Technologies, 23(6), 2251-2265.
10. Gowal, S., Martinoli, A. (2012) Real-time optimization of trajectories that guarantee the rendezvous of mobile robots. IEEE/RSJ International Conference on Intelligent Robots and Systems, 3518-3525.
11. Erunsal, I. K., Ventura R., Martinoli A. (2019) NMPC for 3D Formation of Multirotor MAVs with Relative Sensing in Local Coordinate, arXiv.org, e-print 1904.03742
12. Erunsal, I.K., Martinoli A., Ventura R. (2019) Decentralized NMPC for 3D formation of MAVs with relative sensing and estimation." IEEE International Symposium on Multi-Robot and Multi-Agent Systems, 176-178.
13. Quan Q., (2017) Introduction to multicopter design and control, Ch.6, pp. 134-138, Springer, Beijing,.
14. O. Michel, Webots: Professional mobile robot simulation, International Journal of Advanced Robotic Systems 1 (1) (2004) 39-42.
15. Schiano, F., Franchi, A., Zelazo, D., Giordano, P. R. (2016). A rigidity-based decentralized bearing form. controller for groups of UAVs. IEEE/RSJ International Conference on Intelligent Robots and Systems, 5099-5106.
16. RC Benchmark Series 1585 Thrust Stand, https://www.rcbenchmark.com/,2020
17. Dias, D., Ventura, R., Lima, P., Martinoli, A. (2016). On-board vision-based 3D relative localization system for multiple quadrotors. IEEE International Conferance on Robotics and Automation, 1181-1187.
18. Yuan, Q., Zhan, J. and Li X. (2017). Outdoor flocking of quadcopter drones with decentralized model predictive control. ISA transactions 71, 84-92.
19. Huck, S., Rueppel M., Summers, T., Lygeros, J. (2014). Rcopterx-experimental validation of a distributed leader-follower MPC approach on a miniature helicopter test bed. European Control Conference, 802-807.
20. Gros, S., Zanon, M., Quirynen, R., Bemporad, A., Diehl, M. (2020). From linear to nonlinear MPC: bridging the gap via the real-time iteration. International Journal of Control, 93(1), 62-80.
21. Quirynen, R., Vukov, M., Zanon, M., Diehl, M. (2015). Autogenerating microsecond solvers for nonlinear MPC: a tutorial using ACADO integrators. Optimal Control Applications and Methods, 36(5), 685-704.
22. Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., Diehl, M. (2014). qpOASES: A parametric active-set algorithm for quadratic programming. Mathematical Programming Computation, 6(4), 327-363.