Collision Avoidance for Multiple Micro Aerial Vehicles using Fast Centralized Nonlinear Model Predictive Control ⋆

Björn Lindqvist * **Sina Sharif Mansouri** * **Pantelis Sopasakis** **
George Nikolakopoulos *

** Robotics Team, Department of Computer, Electrical and Space
Engineering, Luleå University of Technology, Luleå SE-97187, Sweden.
** School of Electronics, Electrical Engineering and Computer Science
(EEECS), Queen's University Belfast and Centre for Intelligent
Autonomous Manufacturing Systems (i-AMS), United Kingdom*

Abstract: This article proposes a novel control architecture using a centralized nonlinear model predictive control (CNMPC) scheme for controlling multiple micro aerial vehicles (MAVs). The control architecture uses an augmented state system to control multiple agents and performs both obstacle and collision avoidance. The optimization algorithm used is OpEn, based on the proximal averaged Newton type method for optimal control (PANOC) which provides fast convergence for non-convex optimization problems. The objective is to perform position reference tracking for each individual agent, while nonlinear constraints guarantee collision avoidance and smooth control signals. To produce a trajectory that satisfies all constraints a penalty method is applied to the nonlinear constraints. The efficacy of this proposed novel control scheme is successfully demonstrated through simulation results and comparisons, in terms of computation time and constraint violations, which are provided with respect to the number of agents.

Keywords: Aerial navigation, MAV, Model predictive control, Collision avoidance, Obstructed navigation

1. INTRODUCTION

1.1 Background and Motivation

The deployment of coordinated MAV flights is gaining increasing attention in different applications areas, such as coordinated inspection of infrastructure (Mansouri et al., 2018) or coordinated aerial acrobatics (Mellinger and Kumar, 2011). Such application scenarios call for advanced control algorithms that provide high levels of autonomy with integrated collision avoidance, among the agents and obstacles in the environment.

A centralized control scheme allows all agents to account for obstacles and changes in their environment, as well as the positions of other agents, without agent-to-agent broadcasting. Such a control scheme must provide satisfactory trajectories for all agents, while keeping the computation time low as it is required for the control of MAVs.

Specifically, for a nonlinear model predictive control (NMPC) scheme, the computation time is the greatest bottleneck. But provided that the computation time is constrained low for multiple MAVs, the customizability of the NMPC makes it a great scheme for applying a centralized control. In this article, we present a nonlinear constrained NMPC, based on the proximal averaged Newton method for optimal control (PANOC) (Stella, L. and Themelis, A. and Sopasakis, P. and Patrinos, P., 2017; Small et al., 2019; Sathya et al., 2018) that provides obstacle and collision avoidance for multiple agents using a penalty method (Hermans et al., 2018), while keeping the computation times below a desired bound of 50 ms (Small et al., 2019), which is fast enough to ensure stability of the MAV, but slow enough to have a sufficiently long prediction in the NMPC formulation for obstacle avoidance. A sampling time of 50 ms will thus be the benchmark for solver time in this article.

The problem of path planning (LaValle, 2006) has been widely studied as it a key problem in mobile robotics. In the specific case for MAVs it comes with its own specific challenges, such as the nonlinearity of the system dynamics and the tight runtime requirements, while several approaches have been proposed in the related literature (Goerzen et al., 2010). Popular path planners include the potential fields (Droeschel et al., 2016), graph search methods, such as dynamic variations of A^* (Heng et al., 2011) and model predictive control (Alexis et al., 2011). The overarching goal of such path planners is to have a reactive control system that can provide collision-free paths in constrained environments, while accounting for the nonlinear dynamics of the MAV at a low computation time.

The problem of cooperative path-planning for multiple MAVs can be split into centralized and non-centralized (distributed, decentralized). The distributed and decentralized schemes have been widely used (Montenegro et al., 2015; Kumaresan and Singh, 2016; Richards and How, 2004; Chao et al., 2012), where only specific data are shared among agents. Specifically in regards to a distributed or decentralized MPC, it is a common practice to either share the predicted trajectories (Richards and How, 2004; Chao et al., 2012) or internally predict the positions of other agents using a linear prediction model, to prevent collisions (Kamel et al., 2017a). In such decentralized approaches, individual agents solve lower dimensional optimization problems, thus mitigating the heavy load of solving one large centralized problem. However, according to Negenborn and Hellendoorn (2010) centralized formulations are computationally demanding and outperform their distributed/decentralized counterparts.

A legitimate concern, in the centralized control architectures, is the robustness of the communication of the vehicles to a central computing node. However, with more stable and faster wireless networks, such as 5G and edge computing, a centralized control scheme becomes relevant. In such centralized control schemes, the entire position and orientation information is available to a single computing agent, which can make informed decisions towards a collision-free navigation of all vehicles, while this in turn provides a better closed-loop performance. Centralized schemes are also suitable for small MAVs that do not have adequate computing power. Instead, such computation can, for example, be offloaded to an edge computing system and wirelessly transmitted to the agents (Varghese et al., 2016). Centralized NMPC schemes have been considered before, such as (Erunsal et al., 2019).

All the aforementioned cases of NMPC use some form of SQP (sequential quadratic programming) to solve the optimization problem. The main drawback of SQP lies in the need to solve a quadratic program at each iteration, which requires inner iterations, and the requirement to compute and store Jacobian matrices. The proposed method uses PANOC (Stella et al., 2017; Sathya et al., 2018) as the solver for the centralized control scheme. PANOC uses the same oracle as the projected gradient method (Nesterov, 2018), involves only simple algebraic operations and has a low memory and computational footprint. PANOC has been shown to successfully fly a single MAV in an obstructed environment (Small et al., 2019).

1.2 Contributions

The first contribution of this article is a framework for centralized NMPC (CNMPC) of multiple MAVs. This framework provides collision-free paths, while avoiding obstacles. We consider nonlinear dynamical models for every aerial agent and solve a common optimization problem to decide individual control actions and unlike distributed approaches, we optimize over the trajectories of all agents.

Secondly, we conduct several realistic simulations to evaluate the proposed CNMPC approach in demanding environments (tight formations and obstacles). We analyze the efficacy of CNMPC in terms of the required execution time and show that this scales gracefully with the number
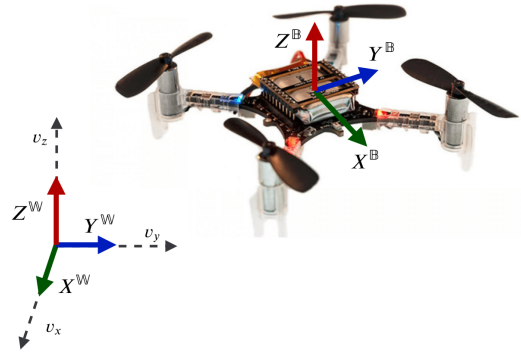

Fig. 1. Utilized coordinate frames, where $\mathbb{W}$ and $\mathbb{B}$ denote the world and body coordinate frames respectively.

of agents. We also evaluate the constraint violations in closed-loop simulations, which are found to be low.

Lastly, we briefly compare the performance of our approach to that of SQP. As it will be demonstrated, the OpEn-based method successfully leads to collision-free paths in all the cases, while the computation time remains within the tight prescribed requirements of 50 ms for up to seven agents.

2. METHODOLOGY

2.1 MAVs Kinematics

The MAV coordinate systems are depicted in Figure 1, where $(x^{\mathbb{B}}, y^{\mathbb{B}}, z^{\mathbb{B}})$ denote the body-fixed coordinate system, while $(x^{\mathbb{W}}, y^{\mathbb{W}}, z^{\mathbb{W}})$ denote the global coordinate system. In this article the states of all agents are defined in a yaw-compensated global frame of reference. The six degrees of freedom (DoF) MAV is defined by the set of equations (1). The full derivation of the adopted model can be found in (Kamel et al. (2017b)).

$$\dot{p}(t) = v(t) \tag{1a}$$

$$\dot{v}(t) = R(\phi,\theta) \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} - \begin{bmatrix} A_x & 0 & 0 \\ 0 & A_y & 0 \\ 0 & 0 & A_z \end{bmatrix} v(t), \tag{1b}$$

$$\dot{\phi}(t) = {}^1\!/\!_{\tau_\phi}(K_\phi \phi_{\text{ref}}(t) - \phi(t)), \tag{1c}$$

$$\dot{\theta}(t) = {}^1\!/\!_{\tau_\theta}(K_\theta \theta_{\text{ref}}(t) - \theta(t)), \tag{1d}$$

where $p = [p_x, p_y, p_z]^\top$ is the position, $v = [v_x, v_y, v_z]^\top$ is the linear velocity in the global frame of reference, and ϕ and $\theta \in [-\pi, \pi]$ are the roll and pitch angles along the $x^{\mathbb{W}}$ and $y^{\mathbb{W}}$ axes respectively. Moreover, $R(\phi(t), \theta(t)) \in \text{SO}(3)$ is a rotation matrix that describes the attitude in Euler form, with $\phi_{\text{ref}} \in \mathbb{R}$, $\theta_{\text{ref}} \in \mathbb{R}$ and $T \geq 0$ to be the references in roll, pitch and the total (massless) thrust generated by the four rotors. The above model assumes that the acceleration depends only on the magnitude and angle of the thrust vector, produced by the motors, as well as the linear damping terms $A_x, A_y, A_z \in \mathbb{R}$ and the gravitational acceleration g.

The attitude terms are modeled as a first-order system between the attitude (roll/pitch) and the references $\phi_{\text{ref}} \in \mathbb{R}$, $\theta_{\text{ref}} \in \mathbb{R}$, with gains $K_\phi, K_\theta \in \mathbb{R}$ and time constants $\tau_\phi, \tau_\theta \in \mathbb{R}$. The aforementioned terms model the closed-loop behavior of a low-level controller, which also implies that the MAV is equipped with a lower-level attitude

controller that takes thrust, roll and pitch commands and provides motor commands for the MAV such as (Jackson et al., 2016).

2.2 Joint Cost Function

Let the state vector of agent i be denoted by $x^{(i)} = [p^{(i)}, v^{(i)}, \phi^{(i)}, \theta^{(i)}]^\top$, and the corresponding control action be $u^{(i)} = [T^{(i)}, \phi_{\text{ref}}^{(i)}, \theta_{\text{ref}}^{(i)}]^\top$. The state variables of all agents can be collected into the vector $x = [x^{(1)}, x^{(2)}, \ldots, x^{(N_a)}]^\top$ where N_a is the number of agents. Likewise, we define the vector of all control actions, $u = [u^{(1)}, u^{(2)}, \ldots, u^{(N_a)}]^\top$.

The system dynamics of all N_a agents are discretized with a sampling time of $T_s = 50\,\text{ms}$ using the forward Euler method to obtain

$$x_{k+1} = f(x_k, u_k). \quad (2)$$

This discrete model is used as the *prediction model* of NMPC. This prediction is done with receding horizon e.g., the prediction considers a set number of steps into the future. We denote this as the *prediction horizon*, N, of the NMPC. By associating a cost to a configuration of states and inputs at the current time and in the prediction, a nonlinear optimizer is tasked with finding an optimal set of control actions, defined by the cost minimum of this *cost function*.

Let $x_{k+j|k}^{(i)}$ denote the predicted state of agent i at time step $k+j$, produced at the time step k. The corresponding control actions are denoted by $u_{k+j|k}^{(i)}$. Let us also denote $\boldsymbol{x}_k = (x_{k+j|k}^{(i)})_{j,i}$ and $\boldsymbol{u}_k = (u_{k+j|k}^{(i)})_{j,i}$. The controller aims to make the state reach the prescribed set points, while delivering smooth control inputs. To that end, we formulate the following cost function:

$$J(\boldsymbol{x}_k, \boldsymbol{u}_k; u_{k-1|k}) = \sum_{j=0}^{N} \sum_{i=1}^{N_a} \underbrace{\|x_{\text{ref}}^{(i)} - x_{k+j|k}^{(i)}\|_{Q_x}^2}_{\text{State cost}}$$
$$+ \underbrace{\|u_{\text{ref}} - u_{k+j|k}^{(i)}\|_{Q_u}^2}_{\text{Input cost}} + \underbrace{\|u_{k+j|k}^{(i)} - u_{k+j-1|k}^{(i)}\|_{Q_{\Delta u}}^2}_{\text{Input smoothness cost}}, \quad (3)$$

where $Q_x \in \mathbb{R}^{8 \times 8}, Q_u, Q_{\Delta u} \in \mathbb{R}^{3 \times 3}$ are symmetric positive definite weight matrices for the states, inputs and input rates respectively. In (3), the first term denotes the *state cost*, which penalizes deviating from a certain state reference x_{ref}. The second term denotes the *input cost* that penalizes a deviation from the steady-state input $u_{\text{ref}} = [g, 0, 0]$ i.e. the inputs that describe hovering. Finally, to enforce smooth control actions, a third term is added that penalizes changes in successive inputs. Note that the first such penalty, $\|u_{k|k}^{(i)} - u_{k-1|k}^{(i)}\|^2$, depends on the previous control action $u_{k-1|k}^{(i)} = u_{k-1}^{(i)}$.

All agents are given the same weight matrices $Q_x, Q_u, Q_{\Delta u}$ and input reference u_{ref}. This approach considers a separate state reference x_{ref} for each agent, which allows for a path-planning approach where each agent can track an individual set-point reference. The Δu-terms $u_{k+j|k} - u_{k+j-1|k}$ are also considered for each agent as the computed control inputs will be different for every agent.

2.3 Input constraints

With the goal of being applicable to a real MAV, hard bounds on reference angles $\phi_{\text{ref}}, \theta_{\text{ref}}$ must be considered as a low-level controller will only be able to stabilize the attitude within a certain range. Since the thrust of a MAV is limited, such hard bounds must also be applied to the thrust input, T. Thus we define bounds on inputs as:

$$u_{\min} \leq u_{k+j|k}^{(i)} \leq u_{\max}, \quad (4)$$

that are considered for all agents in the system, for every step in the prediction.

2.4 Obstacle and collision avoidance

Cylindrical Obstacle   Cylinders, as well as balls and rectangles, are convenient geometrical shapes for enveloping arbitrary obstacles in the environment of the aerial vehicles. Since a cylindrical obstacle can well represent a person or general obstruction, we use this geometry for our obstacle formulation. A cylindrical obstacle can be identified by its position, height and radius, that is, the triplet: $\xi^{\text{obs}} := (p^{\text{obs}}, r^{\text{obs}}, l^{\text{obs}})$. Following Sathya et al. (2018), we define the obstacle avoidance constraint to be:

$$h_{\text{cyl}}(p; \xi^{\text{obs}}) := [p_z - p_z^{\text{obs}} + \tfrac{1}{2} l_{\text{obs}}]_+$$
$$[p_z + p_z^{\text{obs}} + \tfrac{1}{2} l_{\text{obs}}]_+ [r_{\text{obs}}^2 - (p_x - p_x^{\text{obs}})^2$$
$$- (p_y - p_y^{\text{obs}})^2]_+ = 0, \quad (5)$$

where for all $x \in \mathbb{R}$ we define $[x]_+ = \max\{0, x\}$. In other words, the point $p = (p_x, p_y, p_z)$ lies outside the cylinder at hand so long as (5) holds. In the NMPC we shall require that the above constraint is satisfied for all positions of all agents along the prediction horizon.

Obstacles of more general shapes can be considered in a similar fashion following (Small et al., 2019; Sathya et al., 2018).

Collision Avoidance   In a similar fashion, in order to prevent collisions among agents we require that:

$$h_{l,i}(\boldsymbol{x}_k) := [p_{z,k+j|k}^i - p_{z,k+j|k}^l + L]_+ [p_{z,k+j|k}^i$$
$$+ p_{z,k+j|k}^l + L]_+ [r_{\text{safety}}^2 - (p_{x,k+j|k}^{(i)} - p_{x,k+j|k}^{(l)})^2$$
$$- (p_{y,k+j|k}^{(i)} - p_{y,k+j|k}^{(l)})^2]_+ = 0, \quad (6)$$

for all pairs (i, l) with $i, l \in \mathbb{N}_{[1, N_a]}$ and $i < l$, where r_{safety} is the minimum distance that agents l and i should be away from one another. In our NMPC formulation, we shall impose the above constraint between all agents i and l, $i \neq l$ and along the prediction horizon. Note that it is not recommended for the vehicles to fly one above the other, as the top vehicle's wake can have a destabilizing effect on the lower vehicle, thus (6) imposes a high safety distance above each agent.

2.5 Control Input Rate

We impose a constraint on the successive differences of control actions so as to prevent an overly aggressive behavior in the control inputs ϕ_{ref} and θ_{ref}, that is

$$|\phi_{\text{ref},k+j-1|k}^{(i)} - \phi_{\text{ref},k+j|k}^{(i)}| \leq \Delta\phi_{\max}, \quad (7a)$$
$$|\theta_{\text{ref},k+j-1|k}^{(i)} - \theta_{\text{ref},k+j|k}^{(i)}| \leq \Delta\theta_{\max}, \quad (7b)$$

for $j \in \mathbb{N}_{[0,N-1]}$ and $i \in \mathbb{N}_{[1,N_a]}$. The above inequality constraints can be written as equality constraints as follows:

$$[\phi_{\text{ref},k+j-1|k}^{(i)} - \phi_{\text{ref},k+j|k}^{(i)} - \Delta\phi_{\max}]_+ = 0, \qquad (8a)$$

$$[\phi_{\text{ref},k+j|k}^{(i)} - \phi_{\text{ref},k+j-1|k}^{(i)} - \Delta\phi_{\max}]_+ = 0, \qquad (8b)$$

and similarly for θ.

2.6 NMPC and Embedded Optimization

The requirements we outlined above lead to the formulation of the following model predictive control problem for N_a agents and in presence of multiple cylindrical obstacles, identified by ξ_s^{obs} for $s \in \mathbb{N}_{[1,N_o]}$

$$\underset{\boldsymbol{u}_k, \boldsymbol{x}_k}{\text{Min}} \; J(\boldsymbol{x}_k, \boldsymbol{u}_k; u_{k-1|k}) \qquad (9a)$$

$$\text{s.t. } x_{k+j+1|k} = f(x_{k+j|k}, u_{k+j|k}), j \in \mathbb{N}_{[0,N-1]}, \qquad (9b)$$

$$u_{\min} \le u_{k+j|k}^{(i)} \le u_{\max}, j \in \mathbb{N}_{[0,N]}, \qquad (9c)$$

$$h_{\text{cyl}}(p_{k+j|k}^{(i)}; \xi_s^{\text{obs}}) = 0, j \in \mathbb{N}_{[0,N]},$$

$$i \in \mathbb{N}_{[1,N_a]}, s \in \mathbb{N}_{[1,N_o]} \qquad (9d)$$

$$h_{l,i}(\boldsymbol{x}_k) = 0, j \in \mathbb{N}_{[0,N]}, i, l \in \mathbb{N}_{[1,N_a]}, i < l, \qquad (9e)$$

$$\text{Constraints (8)}, j \in \mathbb{N}_{[0,N]}, i \in \mathbb{N}_{[1,N_a]}, \qquad (9f)$$

$$x_{k|k}^{(i)} = x_k^{(i)}, i \in \mathbb{N}_{[1,N_a]}, \qquad (9g)$$

$$u_{k-1|k}^{(i)} = u_{k-1}^{(i)}, i \in \mathbb{N}_{[1,N_a]}. \qquad (9h)$$

This problem fits into the framework of the open-source solver OpEn (Sopasakis et al., 2020), which generates embedded-ready source code written in Rust; a fast programming language that comes with memory safety guarantees. OpEn solves parametric optimization problems of the general form:

$$\text{Minimize}_{z \in Z} \; \ell(z) \qquad (10a)$$

$$\text{subject to: } F(z) = 0, \qquad (10b)$$

where U is a set on which one can easily compute projections, ℓ is a Lipschitz-differentiable function and F is a vector-valued mapping so that $\|F(u)\|^2$ is a Lipschitz-differentiable function.

In order to write Problem (9) in the form of Problem (10), we define the decision variable $z = \boldsymbol{u}_k$, choose Z to be the rectangle defined by the input constraints (9c), eliminate the system dynamics following the single shooting approach of (Small et al., 2019; Sathya et al., 2018) and define F so as to cast all equality constraints. The quadratic penalty method formulates gauge problems (referred to as *inner* problems), which have the form: Minimize$_{z \in Z} \ell(z) + c\|F(z)\|^2$, where c is a positive penalty parameter. The inner problems are solved using PANOC and the penalty parameter is increased in an *outer* iteration loop until $\|F(z)\|_\infty$ drops below a specified *infeasibility tolerance*.

3. SIMULATION RESULTS

3.1 Simulation Model Parameters and Costs

For the presented simulations, the corresponding model parameters can be described as in (1) and are chosen as $\tau_\phi, \tau_\theta = 0.5$, $K_\phi, K_\theta = 1$, in order to approximately match the response of a low-level controller acting on a MAV. Additionally, g is set to $9.82\,\text{m/s}^2$, the control horizon is set

to $N = 30$ which, with a sampling time of $50\,\text{ms}$, implies a prediction of $1.5\,\text{s}$. The weights in (3) are chosen as:

$$Q_x = \text{diag}(5, 5, 20, 3, 3, 3, 8, 8), \qquad (11a)$$

$$Q_u = \text{diag}(5, 10, 10), \qquad (11b)$$

$$Q_{\Delta u} = \text{diag}(10, 25, 25). \qquad (11c)$$

The bounds on control inputs are chosen (in SI units) as:

$$u_{\min} = \begin{bmatrix} 5 \\ -0.4 \\ -0.4 \end{bmatrix}, u_{\max} = \begin{bmatrix} 13.5 \\ 0.4 \\ 0.4 \end{bmatrix}. \qquad (12)$$

Additionally the constraints on change in the input described in (8) are chosen as $\Delta\phi_{\max} = 0.07$ and $\Delta\theta_{\max} = 0.07$.

All simulations use four penalty iterations (except stated otherwise) with initial penalty weight 10 and an update factor of 10 for an exponential increase in costs associated with violating the penalty constraints (obstacle, collision, change in input). In the following simulations we chose obstacles with a large l_{obs} to discourage the MAVs to move above the obstacle.

The state update model is the same as the nonlinear discretized prediction model with an addition of a Gaussian noise parameter. This noise represents a general uncertainty in state data, as well as an uncertainty in how the MAVs behave based on a certain input. Adding noise to the state update forces the optimizer to make realistic micro-adjustments to compensate. The noise parameter is generated with a normal Gaussian distribution with a specified mean, μ, and standard deviation, σ^2 as $\mathcal{N}(\mu, \sigma^2)$ (Peebles, 2001). The noise added to each state are the IID (independent and identically distributed) processes $\eta_p \sim \mathcal{N}(0, 0.01)$, $\eta_v \sim \mathcal{N}(0, 0.005)$ and $\eta_{\theta,\phi} \sim \mathcal{N}(0, 0.001)$, where η_p, η_v, and $\eta_{\theta,\phi}$ are the noise added to the position, velocity, and attitude terms respectively.

3.2 Simultaneous Collision and Obstacle Avoidance

The task of the MAVs in the following simulations is to take off to a set point. At $2.5\,\text{s}$ a new reference set-point is given on the opposite side of a cylindrical obstacle with a radius of $0.8\,\text{m}$ which forces the MAVs to translate past the obstacle while avoiding collisions to arrive at the new set-point.

Figure 2 shows the result by deploying four agents. The application of the collision avoidance constraint is demonstrated by the fact that the trajectory of the outer agents (MAV1 and MAV4) curves out as to satisfy a distance of $0.4\,\text{m}$ to the inner agents (MAV2 and MAV3). The distance between MAVs can be seen in Figure 3. The safety distance is kept to $0.4\,\text{m}$ throughout the avoidance phase, except very small violations at 4 and $12\,\text{s}$ respectively, with a maximum of $0.04\,\text{m}$, which is to be expected due to the noisy state update and limited penalty method iterations. The computation time is found in the same figure and spikes up to a maximum of $27\,\text{ms}$. The insets in the solver time graph shows when the penalty method iterations are added to the computation. Figure 4 shows the control inputs applied to the state update function. The control inputs display a relatively smooth behavior despite the added noise and momentary constraint violations demonstrating the application of (8), and is kept within the control input constraints from (12).
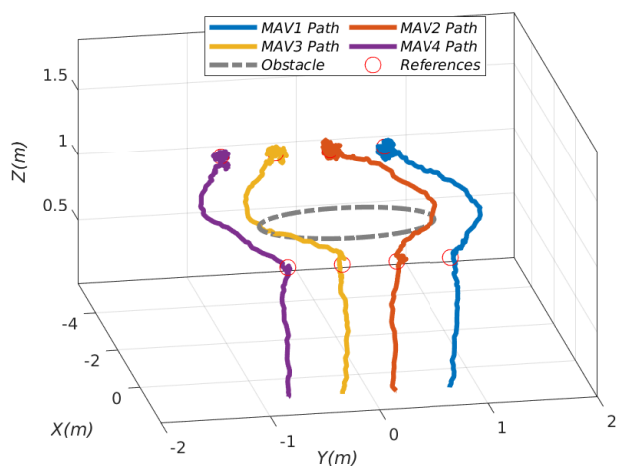
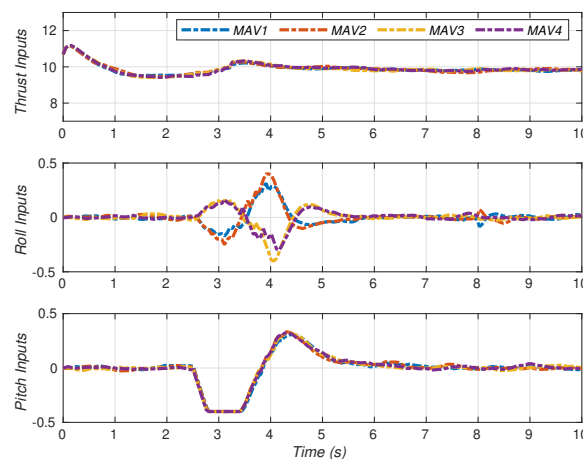Fig. 2. Path of four MAVs avoiding a cylinder with radius 0.8m while keeping a distance of 0.4m.
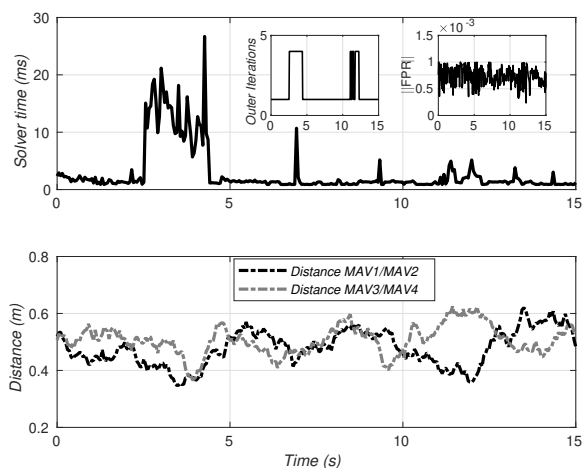

Fig. 3. Solver time (top) and distance between agents (bottom) from four-agent simulation. The two insets in the top figure show the number of outer iterations of the penalty algorithm and the norm of the fixed-point residual for the inner solver, which serves as a measure of the quality of the solution.

3.3 Adding additional MAVs to Collision/Obstacle Avoidance Scenario

The above mission fully demonstrate the essence of the power of the CNMPC. All agents perform individual reference tracking, starting in a tight formation, while avoiding collisions with obstacles and other agents and also keeping the control actions smooth. It is also easy to add additional agents in the obstacle/collision avoidance scenario e.g, the two innermost agents avoid the obstacle and outer agents avoid collisions among each other as the formation is changed due to the obstacle avoidance maneuver. Thus, this configuration will be used as the benchmark test in this article.

The considered factors are the computation time (mean and maximum) as well as the violation of the constraints for both safety distance and obstacles. Due to computation

Fig. 4. Computed control inputs for four-agent simulation. Thrust (top), Roll Reference (middle), Pitch Reference (top).

times restrictions for additional agents the penalty method iterations are forced to be kept low. In such a way computation time is lowered at a cost of losing the guarantee that no constraints are violated which becomes more apparent as more agents are added.

Thus, when using OpEn, as a centralized path planner for multiple agents, the behavior shown in Figure 6, where the maximum constraint violations are shown, is expected. For larger optimization problems, the constraint violations also increase. Table 1 displays the solver times for the simulation scenarios for additional agents, up to a total of nine, where the computation breaks the threshold of 50 ms, with seven and eight agents barely above. Moreover, at 8-9 agents Figure 6 shows a fast increasing violation of the constraints. This, in combination with the increase in solver time, shows the limit of the range of OpEn for this specific configuration of constraints, weights and penalty iterations.

The proposed method show significantly lower computation time compared to centralized and distributed NMPCs solved by Sequential Quadratic Programming (SQP) using `fmincon` in (Mansouri et al., 2015) for the trajectory planning of multiple MAVs with nonlinear models and constraints for safety distance between the MAVs. The problem is solved with prediction horizon of three, sampling time of 0.2 s, and only a one step prediction of Micro Aerial Vehicle (MAV) positions are shared between the agents. In this SQP implementation the mean of computation time reaches to at least 47 s and 16 s with more than two agents for centralized and distributed MPCs respectively.

The comparison with this article is not completely fair, but it does provide a very interesting result that is demonstrated in Figure 5 where additional agents increase both considered solver times linearly, which is the result of a distributed network in Mansouri et al. (2015). In the same article, a centralized scheme results in an exponential increase in solver time for additional agents, again using `fmincon`. Demonstrated here OpEn shows a different behavior as more decision variables and constraints are added in the form of additional agents.

Table 1. Mean and maximum solver time for simulations of 2–9 agents.

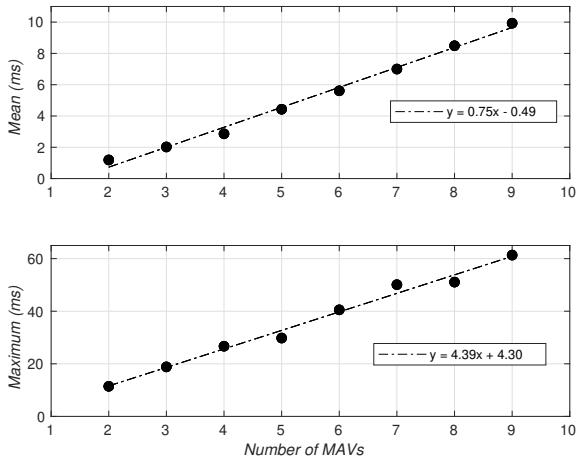| # of MAVs	Solver Time (ms)	
	Mean	Maximum
2	1.19	11.4
3	2.02	18.8
4	2.86	26.7
5	4.43	29.8
6	5.61	40.5
7	7.08	50.0
8	8.49	51.1
9	9.93	61.3

Fig. 5. Minimum solver time (top), Average solver time (middle) and Maximum solver time (bottom) for Obstacle/Collision avoidance simulations.

The aforementioned (Kamel et al., 2017a) and (Erunsal et al., 2019) achieve real-time applicable solver times using decentralized and centralized NMPC, while considering two and three agents respectively, but include no analysis for an increasing number of agents and are also very different in their collision avoidance approach.

3.4 Demanding Collision Avoidance with four MAVs

The previous simulations have all considered a kind of formation flight, where all agents are moving in the same direction and the formation is broken by the obstacle avoidance maneuver. In Figure 7 a scenario where four agents set on a direct collision course with each other is demonstrated. Due to the demanding scenario the penalty method iterations were increased from four to five in this simulation.

In this case, all agents avoid each other, while moving from an initial position to a reference position. Figure 8 shows the distance between agents, as well as the solver time which peaks at the threshold of 50ms for the second computed time step. The fact that all agents almost precisely reach the allowed distance before separating again, while moving to the reference in such a demanding scenario, is a great result for the collision avoidance. This can be attributed to the nature of the centralized
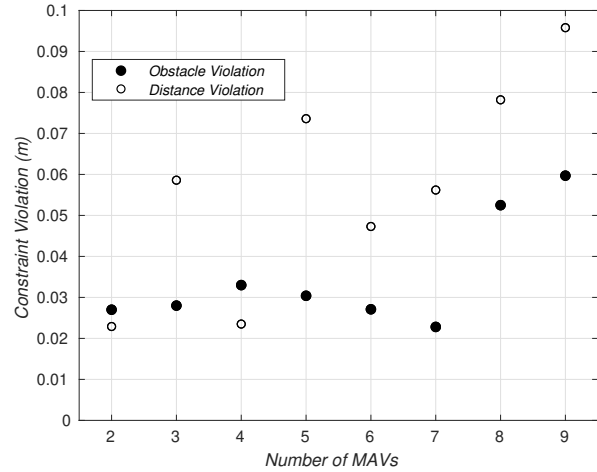

Fig. 6. The maximum violation of obstacle and safety distance constraints for the multiple MAV simulations.
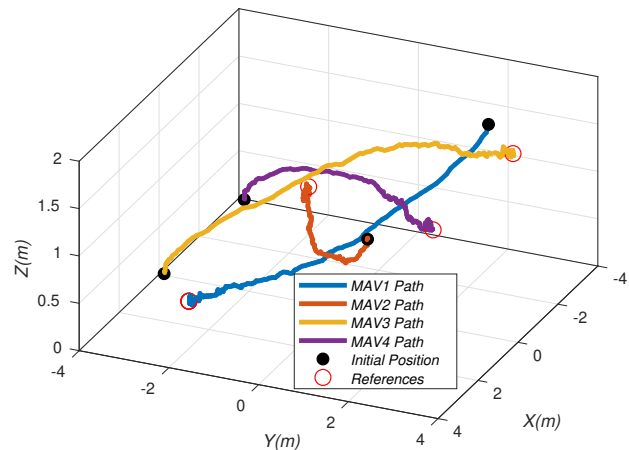

Fig. 7. Paths of 4 MAVs maneuvering in 3D while keeping a distance of 0.4m.

scheme, where the predicted states of all agents are solved for/iterated simultaneously, as to orchestrate trajectories that precisely satisfy the constraints.

4. CONCLUSIONS

This article demonstrated a novel centralized nonlinear model predictive controller for micro aerial vehicles, using OpEn. The proposed scheme accounted for obstacles and collision avoidance in multiple simulation scenarios while keeping the solver time below the considered 50 ms threshold for up to seven agents, while an analysis of solver times and constraint violations are included for up to nine agents performing simultaneous obstacle and collision avoidance maneuvers. In this analysis, it was discovered that the solver time, using OpEn, seems to scale linearly with the number of agents, or decision variables. In the described simulations the OpEn-based CNMPC framework successfully completed the missions of set-point tracking for multiple agents in demanding scenarios while keeping the control signals smooth and without spikes. The limitation of the proposed approach of using a penalty method,
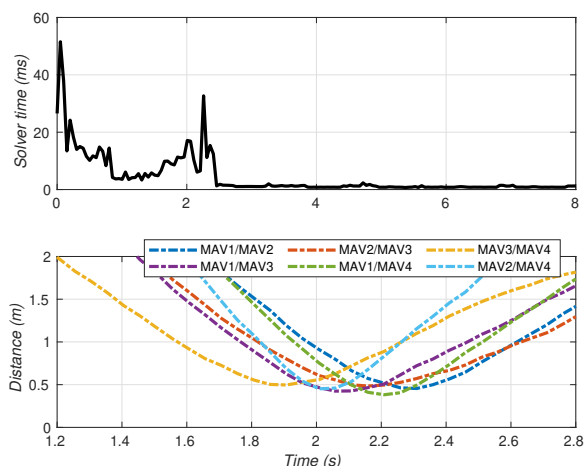
Fig. 8. Solver time (top) and distance between agents (bottom) from collision avoidance simulation with 4 MAVs

with a low number of penalty method iterations to keep the solver times low, is considered, as the constraint violations also increase with an increasing number of agents, and this should be taken into account when applying the CNMPC approach. Future works will include the application this method to laboratory experiments to demonstrate the efficacy of a CNMPC framework for centralized control in a real-world real-time application.

REFERENCES

Alexis, K., Nikolakopoulos, G., and Tzes, A. (2011). Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances. *Control Engineering Practice*, 19(10), 1195–1207.

Chao, Z., Zhou, S.L., Ming, L., and Zhang, W.G. (2012). UAV formation flight based on nonlinear model predictive control. *Mathematical Problems in Engineering*, 2012.

Droeschel, D., Nieuwenhuisen, M., Beul, M., Holz, D., Stückler, J., and Behnke, S. (2016). Multilayered mapping and navigation for autonomous micro aerial vehicles. *Journal of Field Robotics*, 33(4), 451–475.

Erunsal, I.K., Ventura, R., and Martinoli, A. (2019). Nonlinear model predictive control for 3d formation of multirotor micro aerial vehicles with relative sensing in local coordinates. *arXiv preprint arXiv:1904.03742*.

Goerzen, C., Kong, Z., and Mettler, B. (2010). A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent and Robotic Systems*, 57(1-4), 65.

Heng, L., Meier, L., Tanskanen, P., Fraundorfer, F., and Pollefeys, M. (2011). Autonomous obstacle avoidance and maneuvering on a vision-guided MAV using onboard processing. In *IEEE ICRA*, 2472–2477.

Hermans, B., Patrinos, P., and Pipeleers, G. (2018). A penalty method based approach for autonomous navigation using nonlinear model predictive control. *IFAC-PapersOnLine*, 51(20), 234 – 240.

Jackson, J., Ellingson, G., and McLain, T. (2016). ROS-flight: A lightweight, inexpensive MAV research and development tool. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, 758–762.

Kamel, M., Alonso-Mora, J., Siegwart, R., and Nieto, J. (2017a). Nonlinear model predictive control for multi-micro aerial vehicle robust collision avoidance. *arXiv preprint arXiv:1703.01164*.

Kamel, M., Stastny, T., Alexis, K., and Siegwart, R. (2017b). Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system. In *Robot operating system (ROS)*, 3–39.

Kumaresan, G. and Singh, G. (2016). Decentralized formation flying using modified pursuit guidance control laws. In *Indian Control Conference*, 341–346.

LaValle, S.M. (2006). *Planning algorithms*. Cambridge university press.

Mansouri, S.S., Kanellakis, C., Fresk, E., Kominiak, D., and Nikolakopoulos, G. (2018). Cooperative coverage path planning for visual inspection. *Control Engineering Practice*, 74, 118–131.

Mansouri, S.S., Nikolakopoulos, G., and Gustafsson, T. (2015). Distributed model predictive control for unmanned aerial vehicles. In *Workshop on Research, Education and Development of Unmanned Aerial Systems*, 152–161.

Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, 2520–2525. IEEE.

Montenegro, S., Ali, Q., and Gageik, N. (2015). A review on distributed control of cooperating mini UAVs.

Negenborn, R. and Hellendoorn, H. (2010). Intelligence in transportation infrastructures via model-based predictive control. In *Intelligent Infrastructures*, 3–24. Springer.

Nesterov, Y. (2018). *Lectures on convex optimization*, volume 137. Springer.

Peebles, P.Z. (2001). *Probability, random variables, and random signal principles*, volume 3. McGraw-Hill.

Richards, A. and How, J. (2004). Decentralized model predictive control of cooperating UAVs. In *IEEE CDC*, volume 4, 4286–4291.

Sathya, A., Sopasakis, P., Van Parys, R., Themelis, A., Pipeleers, G., and Patrinos, P. (2018). Embedded nonlinear model predictive control for obstacle avoidance using panoc. In *European Control Conference*, 1523–1528.

Small, E., Sopasakis, P., Fresk, E., Patrinos, P., and Nikolakopoulos, G. (2019). Aerial navigation in obstructed environments with embedded nonlinear model predictive control. In *European Control Conference*, 3556–63.

Sopasakis, P., Fresk, E., and Patrinos, P. (2020). OpEn: Code generation for embedded nonconvex optimization. In *IFAC World Congress*. Berlin. Software available at http://doc.optimization-engine.xyz/.

Stella, L., Themelis, A., Sopasakis, P., and Patrinos, P. (2017). A simple and efficient algorithm for nonlinear model predictive control. In *IEEE CDC*, 1939–1944.

Stella, L. and Themelis, A. and Sopasakis, P. and Patrinos, P. (2017). A simple and efficient algorithm for nonlinear model predictive control.

Varghese, B., Wang, N., Barbhuiya, S., Kilpatrick, P., and Nikolopoulos, D.S. (2016). Challenges and opportunities in edge computing. In *SmartCloud*, 20–26.