A Scalable Distributed Collision Avoidance Scheme for Multi-agent UAV systems

Björn Lindqvist[1], Pantelis Sopasakis[2] and George Nikolakopoulos[1].

Abstract— In this article we propose a distributed collision avoidance scheme for multi-agent unmanned aerial vehicles (UAVs) based on nonlinear model predictive control (NMPC), where other agents in the system are considered as dynamic obstacles with respect to the ego agent. Our control scheme operates at a low level and commands roll, pitch and thrust signals at a high frequency, each agent broadcasts its predicted trajectory to the other ones, and we propose an obstacle prioritization scheme based on the shared trajectories to allow up-scaling of the system. The NMPC problem is solved using an *ad hoc* solver where PANOC is combined with an augmented Lagrangian method to compute collision-free trajectories. We evaluate the proposed scheme in several challenging laboratory experiments for up to ten aerial agents, in dense aerial swarms.

A. Introduction and Background

One of the most popular and exciting areas of robotics right now is the area of collaborative robotics, meaning that a team of robots are tasked with collaboratively performing a specified mission, which could include collaborative inspection[1], mapping & exploration[2], search-and-rescue[3] and many others. The co-operation and coordination of teams of robots can be done in many different ways, but a common nomenclature is the division into centralized, decentralized, or distributed schemes. For centralized approaches all computation and planning is done via a single computational agent, which is fed all available system information. In the decentralized approach, all agents act independently only based on information available to that agent and as such allow for much greater scalability. The distributed scheme is the middle ground, where every agent computes its own decisions, but specific information is transmitted between agents that supplement the computed decisions, which allows for both great co-operation and scalability.

When there are multiple robotic agents occupying a small space to perform a task, the collision avoidance scheme must not only avoid collisions with the environment, but also collision among agents. In this article we propose a distributed nonlinear model predictive control (DNMPC), where computed trajectories are shared among agents, and other agents in the system are considered as obstacle constraints.

There are many different approaches to collision avoidance schemes for multi-agent robotic systems. Rule-based schemes, such as potential functions [4], [5] or optimal control schemes

[6], work well in low-density swarms of agents but do not include hard guarantees on safety distances or to do so must include additional restrictive rules. A differential game approach is proposed in [7] with impressive results, but relies on many conditions, and the article only offers simulation results. In [8] a mixed-integer quadratic program is proposed for centralized trajectory generation, but does not allow for real-time solutions meaning it is non-reactive. Other modern solutions showing promise are the barrier functions [9], or *safety barrier certificates* which were evaluated for mini quad-copters in laboratory experiments, and [10] that combined potential-like functions with adaptive control to achieve collision avoidance robust to model uncertainty.

Model predictive control (MPC) schemes for multi-agent collision avoidance are, due to their high performance, flexibility, and ability to handle constraints, one of the most popular approaches to multi-agent collision avoidance [11]. The ability to optimize over future predicted states allows for a direct consideration of moving obstacles, and with fast optimization algorithms allow for real-time reactive and proactive avoidance. MPC schemes comes in many different flavors: in terms of the system dynamics and constraints there are linear [12] and nonlinear [13] solutions, and in terms of architecture there are decentralized [14], centralized [15], and distributed [16] schemes.

In [12], a linear UAV model is used where the decision variable of the MPC is a position-trajectory e.g. it requires a separate position-reference tracking controller to follow the computed trajectory. All agents share predicted trajectories in a distributed fashion, show great scalability, and was experimentally evaluated for up to 20 agents.

In [14] a NMPC scheme is used for multi-agent decentralized collision avoidance. No information is shared among agents, and predicted future positions of the non-ego agents are done with a constant velocity model from initial position and velocity measurements. The motion intent of other agents is simplified, but no information transfer among agents is required. However, the experimental validation involves only two UAVs.

In [17] a method proposing probabilistic collision avoidance for multi-agent systems using NMPC is demonstrated, and shows experimental results for low-density and low numbers of robots, while up-scaling is only demonstrated in simulation.

In our previous paper [15] we proposed a centralized NMPC scheme, where a single computational agent optimized the trajectories of all agents. While this scheme showed good results in simulation, all centralized schemes will eventually

[1] The authors are with the Robotics and AI Team, Department of Computer, Electrical and Space Engineering, Luleå University of Technology, Luleå SE-97187, Sweden

[2] The author is with the School of Electronics, Electrical Engineering and Computer Science (EEECS), Queen's University Belfast and Centre for Intelligent Autonomous Manufacturing Systems (i-AMS), United Kingdom

Corresponding Author's email: `bjolin@ltu.se`

suffer from scalability issues.

A common limitation of existing MPC approaches is that since the number of constraints (or potential-like terms) of the underlying optimization problem must remain constant during runtime, there can be situations where there are too many agents in proximity to the ego vehicle. Moreover, selecting the closest neighbors (as in [12]) may disregard those — potentially more remote — agents that are on a direct collision course with the ego agent.

B. Contributions

We propose a distributed NMPC solution where the collision avoidance is solved for in the control layer, and as such no additional position- or velocity-tracking controller is needed. In our approach, each agent uses a nonlinear dynamical model and shares its predicted trajectory with all neighbouring agents at every sampling time. Since these trajectories are produced by a nonlinear model, they are sufficiently close to their actual trajectories to allow for collision-free coordination. Furthermore, we propose an obstacle prioritization algorithm that determines which agents are on the most dangerous predicted potential collision courses with the ego agent based on their shared predicted motion. Additionally we propose an adaptive scheme for the MPC weights that facilitates challenging collision avoidance situations. This allows us to sacrifice the speed of reaching the target for enforcing collision avoidance. This leads to a highly scalable paradigm that can accommodate large numbers of agents without compromising computational feasibility.

In our proposed scheme, we use a tailored method that has been implemented in Optimization Engine (for short `OpEn`),which is an open-source code generation software for embedded nonlinear optimization [18], [19], that is fully ROS-integrated. It generates Rust code, which is very fast and provably memory safe. `OpEn` uses PANOC (proximal averaged Newton-type method for optimal control) [20], [21] combined with an augmented Lagrangian method [18], [22] to account for general non-convex constraints such as the ones that result from collision avoidance. To the best of our knowledge, this is the first work that offers experimental results for collision avoidance using the augmented Lagrangian method and we shall demonstrate that the proposed scheme is suitable for a fast real-time implementation.

We present results from multiple experimental scenarios for up to ten agents in tight formations, and with low numbers of constraints to reduce computational complexity, to demonstrate the collision avoidance capabilities of the proposed control architecture.

I. METHODOLOGY

A. System Dynamics

The adopted system model is a nonlinear dynamic UAV model, successfully used in previous applications of NMPC for UAVs [15], [23], [24], and thus we will not go into much detail in this article. Importantly, it considers eight states namely: positions coordinates $p = [p_x, p_y, p_z]^\top$, linear velocities $v = [v_x, v_y, v_z]^\top$ as well as roll and pitch angles ϕ

and $\theta \in [-\pi, \pi]$. $\dot{\phi}$ and $\dot{\theta}$ are modeled as first order systems to approximate the closed loop behavior of an attitude controller with inputs $\phi_{\text{ref}}, \theta_{\text{ref}}$, and as such the control inputs of the system are $T, \phi_{\text{ref}}, \theta_{\text{ref}} \in \mathbb{R}$ where $T \geq 0$ is the total mass-less thrust produced by the motors. Based on this model let us define the state vector as $x = [p, v, \phi, \theta]^\top$ and the control actions as $u = [T, \phi_{\text{ref}}, \theta_{\text{ref}}]^\top$. The full dynamic model is as follows:

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

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

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

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

This model is then discretized by the forward Euler transformation to achieve the predictive form

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

This model is used as the prediction model for the receding horizon NMPC-problem, where the number of predicted time instants is denoted by the prediction horizon, N.

B. Control Objective

The main control objective of each agent is to track a given set point, while avoiding collisions with other agents. Additional objectives are not to have abruptly changing control actions and to keep the control actions within certain bounds. Here, we will translate these requirements into a cost function and constraints for a nonlinear MPC formulation.

1) Objective Function: Let $x_{k+j|k}$ and $u_{k+j|k}$ denote the predicted state and input at time $k+j$, computed at the time k. Let $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ be the vectors of all predicted states and inputs along the prediction horizon. The deviation of the predicted state from the set point, x_{ref}, can be penalized using a standard quadratic function and, likewise, we introduce a quadratic cost the input deviation. Additionally, we introduce a penalty on successive changes in control inputs, $u_{k+j|k} - u_{k+j-1|k}$ (note that $u_{k-1|k} = u_{k-1}$ which is the the previous control action), and a standard quadratic terminal state cost. The overall cost is given by

$$J(\boldsymbol{x}_k, \boldsymbol{u}_k, u_{k-1|k}) = \sum_{j=0}^{N-1} \big(\underbrace{\|x_{\text{ref}} - x_{k+j|k}\|_{Q_x}^2}_{\text{State penalty}}$$
$$+ \underbrace{\|u_{\text{ref}} - u_{k+j|k}\|_{Q_u}^2}_{\text{Input penalty}} + \underbrace{\|u_{k+j|k} - u_{k+j-1|k}\|_{Q_{\Delta u}}^2}_{\text{Input change penalty}} \big)$$
$$+ \underbrace{\|x_{\text{ref}} - x_{k+N|k}\|_{Q_t}^2}_{\text{Terminal state penalty}}, \tag{3}$$

where $Q_x, Q_t \in \mathbb{R}^{8 \times 8}, Q_u, Q_{\Delta u} \in \mathbb{R}^{3 \times 3}$ are positive definite weight matrices for the states, terminal state, inputs and input change respectively.

2) Obstacle Definition: We require that the ego agent does not approach any of the other agents at a distance closer than a safety value r_{obs}. Let p and $p^{\mathrm{obs}} = [p_x^{\mathrm{obs}}, p_y^{\mathrm{obs}}, p_z^{\mathrm{obs}}]$ denote the positions of the ego agent and a non-ego agent respectively. Then, define

$$
\begin{aligned}
h_{\mathrm{sphere}}(p, \xi^{\mathrm{obs}}) = (r^{\mathrm{obs}})^2 &- (p_x - p_x^{\mathrm{obs}})^2 \\
&- (p_y - p_y^{\mathrm{obs}})^2 - (p_z - p_z^{\mathrm{obs}})^2, \quad (4)
\end{aligned}
$$

with $\xi^{\mathrm{obs}} = [p^{\mathrm{obs}}, r^{\mathrm{obs}}]$. The obstacle avoidance requirement is equivalent to $h_{\mathrm{sphere}} \leq 0$, that ego agent position p is required to lie completely outside of the sphere defined by ξ^{obs}. Additionally, we require that the constraint holds for predicted positions of the ego vehicle, $p_{k+j|k}$, and predicted obstacle positions $p_{k+j|k}^{\mathrm{obs}}$ along the prediction horizon.

3) Input bounds: For a UAV system based on (1), an attitude controller will only be able to stabilize the UAV within a specific range of ϕ and θ, therefore the control actions commanded by the NMPC and especially ϕ_{ref} and θ_{ref}, should be constrained. Additionally, to further limit the acceleration of the UAV, we also place such limits on T. For this purpose we impose the bounds

$$
u_{\min} \leq u_{k+j|k} \leq u_{\max}. \quad (5)
$$

C. Obstacle Prioritization

An important concept in multi-agent system is scalability. Assuming a system composed of N_a agents, ideally each agent should be able to form obstacle constraints with all other agents, such that the number of obstacles $N_{\mathrm{obs}} = N_a - 1$. Due to limitation in computation power and the speed of optimization algorithms, this is not always possible for high numbers of agents. Instead, we need to choose $N_{\mathrm{obs}} < N_a - 1$ necessitating some kind of obstacle prioritization where the ego agent takes into account a limited number of N_{obs} other agents. Assuming access to the predicted trajectories, $u_{k-1}^{\mathrm{obs},i}$ and measured states $\hat{x}_k^{\mathrm{obs},i}$ of nearby agents, we want the prioritization scheme to be fully based on the motion intentions of each agent. For this purpose we propose Algorithm 1 as the obstacle prioritization scheme, which can be described as follows:

- Using the NMPC prediction model (2) we can describe any $x_{k-1+j|k-1}$ from u_{k-1} and $\hat{x}_k$, and similarly using the shared NMPC solutions $u_{k-1}^{\mathrm{obs},i}$ and $\hat{x}_k^{\mathrm{obs},i}$ to describe $x_{k-1+j|k-1}^{\mathrm{obs},i}$ for $i = 1 \ldots N_a - 1$.
- Calculate the predicted Euclidean distances between the ego agent and all other agents at the current and future time instants $j = 0, \ldots, N$
- An agent is prioritized if the distance is below a threshold specified by $r^{\mathrm{obs},i} + d_s$ where d_s is a positive safety distance. This is done via the following weighted sum as a gauge for the prioritization

$$
w_i = \sum_{j=0}^{N} \alpha(d_{i,j}, v_{k-1+j|k-1}^{\mathrm{obs},i}) \beta(j),
$$

where $d_{i,j}$ denotes the distance to the *i-th* agent at time instant j. Let $\alpha(d_{i,j}, v_{k-1+j|k-1}^{\mathrm{obs},i})$ and $\beta(j)$ be

decreasing functions in $d_{i,j}$ and j respectively, and as such the scheme prioritizes agents that are at a closer predicted distance, and at less distant predictions.
- We also add an extra safety protocol by adding a large number M to w_i if the agents are closer than the obstacle radius $r^{\mathrm{obs},i}$ at the current measured positions ($j = 0$) to always prioritize agents that directly violate the obstacle constraint at the current time instant.
- Obstacle trajectories $(\xi_j^{\mathrm{obs},i})_{i,j}$ are then sorted by the corresponding values in w in descending order, to prioritize the N_{obs} trajectories that produced the largest sums w_i.

For the weight functions we propose simple expressions with the desired functionality, and in this article we are using $\alpha(d, v) = (1 - \frac{d}{r^{obs}+d_s})^2 \|v\|$ (velocity compensation since faster moving obstacles spend fewer time instants in $r^{\mathrm{obs},i} + d_s$) and $\beta(j) = \frac{N}{(j+1)^a}$ with a being a tuning constant to describe the relative emphasis on closer versus more distant time instants.

Algorithm 1: Obstacle Prioritization

Inputs: $\hat{x}_k, u_{k-1}, \hat{x}_k^{\mathrm{obs},i}, u_{k-1}^{\mathrm{obs},i}, N_a, N_{\mathrm{obs}}, r^{\mathrm{obs}}, d_s$

Result: From the shared trajectories and measured states decide which N_{obs} agents should be considered as obstacles

for $i = 1, N_a - 1$ **do**
    for $j = 0, N$ **do**
        Compute $p_{k-1+j|k-1}, p_{k-1+j|k-1}^{\mathrm{obs},i}$ and $v_{k-1+j|k-1}^{\mathrm{obs},i}$
        $d \leftarrow \|p_{k-1+j|k-1} - p_{k-1+j|k-1}^{\mathrm{obs},i}\|$
        $v_{\mathrm{m}} \leftarrow \|v_{k-1+j|k-1}^{\mathrm{obs},i}\|$
        if $(d \leq r^{\mathrm{obs},i})$ *and* $(j = 0)$ **then**
            $w_i \leftarrow w_i + M$
        else if $d \leq r^{\mathrm{obs},i} + d_s$ **then**
            $w_i \leftarrow w_i + (1 - \frac{d}{r^{\mathrm{obs}}+d_s})^2 v_{\mathrm{m}} \frac{N}{(j+1)^a}$
Sort in descending order: $(\xi_j^{\mathrm{obs},i})_{i,j}$ by corresponding element in w_i
$(\xi_{\mathrm{prio},j}^{\mathrm{obs},i})_{i,j} \leftarrow [(\xi_j^{\mathrm{obs},1})_j, (\xi_j^{\mathrm{obs},2})_j, \ldots, (\xi_j^{\mathrm{obs},N_{\mathrm{obs}}})_j]$
Output: $(\xi_{\mathrm{prio},j}^{\mathrm{obs},i})_{i,j}$

D. Embedded Model Predictive Control

1) NMPC Problem: In light of the aforementioned objectives and constraints, the obstacle avoidance problem for each agent leads to the following constrained nonlinear optimal control problem

$$
\underset{u_k, x_k}{\text{Minimize}} \; J(x_k, u_k, u_{k-1|k}) \tag{6a}
$$

$$
\text{subj. to: } x_{k+j+1|k} = \zeta(x_{k+j|k}, u_{k+j|k}), j \in \mathbb{N}_{[0,N-1]}, \tag{6b}
$$

$$
u_{\min} \leq u_{k+j|k} \leq u_{\max}, j \in \mathbb{N}_{[0,N-1]}, \tag{6c}
$$

$$
h_{\mathrm{sphere}}^i(p_{k+j|k}, \xi_{\mathrm{prio},j}^{\mathrm{obs},i}) \leq 0, \; j \in \mathbb{N}_{[0,N]}, \tag{6d}
$$

$$
i \in \mathbb{N}_{[1,N_{\mathrm{obs}}]}, \tag{6e}
$$

$$
x_{k|k} = \hat{x}_k, \tag{6f}
$$

where $\hat{x}_k$ is the current estimated system state. This problem needs to be solved at every sampling time by each agent taking into account the trajectories, $(\xi_j^{\text{obs},i})_{i,j}$, that the other agents have shared. The problem yields an optimal sequence of control actions, the first one of which is applied to the ego vehicle and the optimal predicted trajectory is broadcast to all surrounding agents.

2) Embedded numerical optimization: The optimization problem in Equation (6) can be written concisely in the following form

$$\mathbb{P}(x_{k|k}) : \underset{\boldsymbol{u}_k \in U}{\text{Minimize}}\, f(\boldsymbol{u}_k; x_{k|k}) \tag{7a}$$

$$\text{subject to: } F(\boldsymbol{u}_k; x_{k|k}) \leq 0, \tag{7b}$$

where $f(\,\cdot\,; x_{k|k}) : \mathbb{R}^{3N} \to \mathbb{R}$ is a Lipschitz-differentiable function and $F(\,\cdot\,; x_{k|k}) : \mathbb{R}^{3N} \to \mathbb{R}^{NN_{\text{obs}}}$ is a differentiable mapping with Lipschitz-continuous Jacobian. Here the optimization is carried out over sequences of control actions, $\boldsymbol{u}_k$, while the sequence of states, $\boldsymbol{x}_k$, has been eliminated following what is known as the *sequential* or *single shooting* formulation [24]. The cost function in (7a) is defined by (3), where the state sequence has been eliminated and the constraints in Equation (7b) correspond to the obstacle avoidance constraints discussed in Section I-B.2. The sequence of control actions, $\boldsymbol{u}_k$, is constrained in a set $U \subseteq \mathbb{R}^{3N}$, which is the rectangle defined by the constraints of Equation (5).

Problem $\mathbb{P}(x_{k|k})$ in (7) is solved by using the augmented Lagrangian method. The associated augmented Lagrangian function is

$$L_c(\boldsymbol{u}_k, \boldsymbol{v}_k, \boldsymbol{y}_k; x_{k|k}) = f(\boldsymbol{u}_k; x_{k|k})$$
$$+ \boldsymbol{y}_k^\top \left(F(\boldsymbol{u}_k; x_{k|k}) - \boldsymbol{v}_k \right) + \tfrac{c}{2}\|F(\boldsymbol{u}_k; x_{k|k}) - \boldsymbol{v}_k\|^2, \tag{8}$$

defined for $\boldsymbol{v}_k \in U$ and $\boldsymbol{v}_k \leq 0$ It has been shown in [18] that

$$\min_{\boldsymbol{u}_k \in U, \boldsymbol{v}_k \leq 0} L_c(\boldsymbol{u}_k, \boldsymbol{v}_k, \boldsymbol{y}_k; x_{k|k}) = -\tfrac{1}{2c}\|\boldsymbol{y}_k\|^2$$
$$+ \min_{\boldsymbol{u}_k \in U} \psi(\boldsymbol{u}_k; c, \boldsymbol{y}_k, x_{k|k}), \tag{9}$$

where ψ is defined as

$$\psi(\boldsymbol{u}_k; c, \boldsymbol{y}_k, x_{k|k}) = f(\boldsymbol{u}_k; x_{k|k})$$
$$+ \tfrac{c}{2}\left\| F(\boldsymbol{u}_k; x_{k|k}) + \tfrac{1}{c}\boldsymbol{y}_k - [F(\boldsymbol{u}_k; x_{k|k}) + \tfrac{1}{c}\boldsymbol{y}_k]_- \right\|^2,$$

where $[z]_- = \max\{0, -z\}$. Function ψ is continuously differentiable and has a Lipschitz-continuous gradient, therefore the "inner" optimization problem $\min_{\boldsymbol{u}_k \in U} \psi$ can be solved very efficiently using PANOC. The gradient of ψ can be determined by means of automatic differentiation. This leads to Algorithm 2.

The most important tuning parameter of the algorithm is the penalty update parameter $\rho > 1$. Typical values that seem to work well are between 1.1 and 5. A value close to 1 will make increase the penalty parameter slowly, therefore, the optimal solution $\boldsymbol{u}_k^\nu$ at iteration ν will be a good initial guess for the inner optimization problems, which are expected to

---

Algorithm 2: Augmented Lagrangian method for solving $\mathbb{P}(x_{k|k})$

Inputs: $\boldsymbol{u}_k^0 \in \mathbb{R}^{3N}$ (initial guess), $x_{k|k} \in \mathbb{R}^8$ (parameter), $\boldsymbol{y}_k^0 \in \mathbb{R}^{NN_{\text{obs}}}$ (initial guess for the Lagrange multipliers), $\epsilon, \delta > 0$ (tolerances), ρ (penalty update coefficient), c_0 (initial penalty), θ (sufficient decrease coefficient), $M \gg 1$ (large constant)

Result: (ϵ, δ)-approximate solution $(\boldsymbol{u}_k^\star, \boldsymbol{y}_k^\star)$

for $\nu = 0, \ldots, \nu_{\max}$ **do**
  $\bar{\boldsymbol{y}}_k^\nu = \max\{0, \min\{M, \boldsymbol{y}_k^\nu\}\}$
  Compute a solution
  $\boldsymbol{u}^{\nu+1} \in \arg\min_{\boldsymbol{u}_k \in U} \psi(\boldsymbol{u}_k; c_\nu, \bar{\boldsymbol{y}}_k^\nu, x_{k|k})$ with tolerance $\bar{\epsilon}$ and initial guess $\boldsymbol{u}^\nu$ using PANOC
  Update $\boldsymbol{y}_k^{\nu+1}$ by

$$\begin{aligned} \boldsymbol{y}_k^{\nu+1} = \bar{\boldsymbol{y}}_k^{\nu+1} &+ c_\nu(F(\boldsymbol{u}^{\nu+1}; x_{k|k}) \\ &- \left[F(\boldsymbol{u}^{\nu+1}; x_{k|k}) + c_\nu^{-1}\bar{y}^\nu \right]_-) \end{aligned} \tag{10}$$

  $z_{\nu+1} = \|\boldsymbol{y}_k^{\nu+1} - \boldsymbol{y}_k^\nu\|_\infty$
  if $z_{\nu+1} \leq c_\nu\delta$ *and* $\bar{\epsilon}_\nu \leq \epsilon$ **then**
    return $(\boldsymbol{u}_k^\star, \boldsymbol{y}_k^\star) = (\boldsymbol{u}_k^{\nu+1}, \boldsymbol{y}_k^{\nu+1})$
  else if $\nu > 0$, $z_{\nu+1} > \theta z_\nu$ **then**
    $c_{\nu+1} = \rho c_\nu$
  $\bar{\epsilon}_{\nu+1} = \tfrac{1}{2}\bar{\epsilon}_\nu$

---

converge faster, but this will come at the expense of more ALM iterations. On the other hand, a larger value of ρ will lead to a lower number of ALM iterations, but the initial guess for the inner problems will not be as good, this requiring a higher computational effort.

Overall, the algorithm has low memory requirements, involves only simple operations (no linear systems or factorisations) and is very fast. Upon termination returns an ϵ-suboptimal and δ-infeasible solution.

The algorithm is implemented in OpEn which generates Rust code that solves the parametric problems $\mathbb{P}(x_{k|k})$ for each agent [18], [19].

E. Adaptive NMPC Weights

The optimal Lagrange multipliers, $\boldsymbol{y}_k^\star$, can be thought as indicators of how much the optimal trajectories need to "bend" to avoid the obstacles. The idea is to use $\boldsymbol{y}_k^\star$ to update the reference tracking weights so that obstacle avoidance is prioritized over set point tracking.

Let Q_p define the first three diagonal elements of positive-definite weight matrix Q_x. We introduce a scaling factor that adapts Q_p from $Q_{p,\min}$ to $Q_{p,\max}$ based on Lagrange multiplier $\boldsymbol{y}_k^\star$ as follows:

$$Q_p = Q_{p,\min} + \frac{Q_{p,\max} - Q_{p,\min}}{\sum_{l=0}^{N_{\text{obs}}N} W_l y_{k,l}^* + 1} \tag{11}$$

where W_l is some weight that is decreasing with respect to elements in $y_{k,l}^*$ that represents constraints at more distant future time instants, which in our formulation results in $W_l =$

$b(1 - \frac{l \mod N}{N})$ where b is a tuning constant. This heuristic seems to work well in practise. All elements in Q_p are scaled by the same factor, but it is enough for reducing the general emphasis on reference tracking. Note that the terminal state penalty still promotes the UAV to be as close to its end goal as possible but only at $j = N$.

II. RESULTS

A. Experiment Set-up and NMPC Tuning

The platform used for the experiments is the Crazyflie Nano 2.0 [25], using a ros-stack developed for the Crazyflie [26] for communication with the agents and low-level attitude control. These are small lightweight platforms that do not carry their own computational devices (which does imply some communicated delays), but are great for showing scalability. As such, computation for all agents is done on a single Lenovo ThinkPad T490, with a 1.8GHz Core i7 CPU, with controllers for each UAV running as separate ROS-nodes [27]. All state measurements $\hat{x} = [\hat{p}, \hat{v}, \hat{\phi}, \hat{\theta}]^\top$ are made by a Vicon motion-capture system and a 3-point median filter with outlier rejection to estimate velocities from Vicon position measurements.

The proposed control structure has a large number of model and tuning parameters; model parameters are $A_x = 0.1, A_y = 0.1, A_z = 0.2, K_\theta, K_\phi = 1$ and $T_\phi, T_\theta = 0.5$. For the objective function weight matrices are chosen as $Q_x = \text{diag}(Q_{p,1}, Q_{p,2}, Q_{p,3}, 6, 6, 6, 8, 8)$ with, according to (11), $Q_{p,min} = \text{diag}(1, 1, 15)$ and $Q_{p,max} = \text{diag}(6, 6, 45)$, $Q_u = \text{diag}(5, 10, 10)$, $Q_{\Delta u} = \text{diag}(10, 20, 20)$ and lastly $Q_t = \text{diag}(40, 40, 150, 20, 20, 30, 30)$. The NMPC prediction horizon is set to $N = 40$ with a sampling time of $50ms$, which implies a prediction of two seconds. Adaptive weight tuning constant b is set to 0.01, and j-scaling constant a in the prioritization scheme is set to 0.7, and $d_s = 0.2$ m.

The NMPC constraints described by (5) are $u_{min} = [5, -0.25, -0.25]$ and $u_{max} = [12.5, 0.25, 0.25]$, while obstacle radii are $r^{obs} = 0.4$ m. Note that small inaccuracies in localization, model mismatch, inherent delays in the system and solver tolerances will always prevent this distance from being perfectly maintained. This should be compensated by increasing the obstacle radii above the safety-critical condition which for our choice of platform is around 0.3 m.

Lastly, the number of parametric spherical obstacles in the NMPC formulation is set to $N_{obs} = 3$, while the penalty update coefficient $\rho = 1.5$ and initial penalty $c_0 = 1000$.

B. Experimental Validation Results

To evaluate the proposed distributed collision avoidance method, we present a series of challenging laboratory experiments in which a team of UAVs performs individual position reference tracking in tights formations and challenging collision avoidance scenarios. We include up to ten agents and in the final experiment we also present the addition of a non-cooperative UAV agent to the system. Figure 4 shows the minimum agent-agent distances during all three experiments, while Figure 5 includes histograms of solver times of the NMPC module. For visualizing the experiments

and seeing the real-time behavior of the agents we strongly suggest the reader to watch the experiment video found at **https://youtu.be/3kyiL6MZaag**.

1) Position swapping in tight formations: Figure 1 shows the experiment set-up. The task is for the agents to hold the formation while every five seconds for one minute one agent moves to fill the unoccupied spot. The obstacle prioritization scheme allows the moving agent to quickly couple with the relevant agents as to move through the formation without collisions. Additionally, due to the sharing of the motion intentions directly though the NMPC trajectory, the formation can (mostly) be maintained as other agents see that no collision is imminent.

The minimum agent-agent distance during the experiment was 0.38 m. Due to the complexity and large horizon of the NMPC problem, the optimizer rarely (0.03% of instances) did not converge to the specified tolerance within the bounds for solver time, set at 40 ms to never have run-time issues, and the non-converged solution is instead applied to the system. The average solver time of all agents through the experiment was 1.46 ms.

2) Avoidance with multiple simultaneously moving agents: A more challenging scenario in terms of collision avoidance is when there are multiple moving agents on direct collision courses. Figure 2 shows the set-up, where two teams of five agents are tasked to swap positions with the opposing team while maintaining collision-free paths, repeated two times. This set-up challenges the obstacle prioritization, as the closest agents are not the ones on a collision course.

The agents successfully swap positions with a minimum distance of 0.37 m. Figure 5 shows a higher average solver time due to the shorter but more intensive experiment. The average solver time was 5.35 ms, with around 0.7% of instances reaching the maximum allowed solver time of 40 ms.

Figure 6 displays solver data for one of the agents during the experiment: inner iterations, the suboptimality described by the norm of the fixed-point residual seen as a measure for the quality of the solution, the norm of the vector of Lagrange multipliers ,$\|y_k^*\|$, as well as the δ-infeasibility. The suboptimalty is kept at the specified solver tolerance of 10^{-4}. As is expected the Lagrange Multipliers and δ-infeasibility see a spike as the avoidance maneuvers are initiated and then rapidly drop back to zero.

3) Introducing a non-cooperative obstacle to the system: While the distributed scheme works very well for maintaining the desired agent-agent distance, let us investigate the addition of a non-cooperative obstacle in the form of a manually controlled UAV. We use the common [14], [28] constant velocity model ($\dot{p}^{obs,nc} = v^{obs,nc}$) for moving obstacles. Discretizing the linear model we can predict obstacle positions $p_{k+j|k}^{obs,nc}$ from the measured state $\hat{x}^{obs,nc} = [\hat{p}^{obs,nc}, \hat{v}^{obs,nc}]$ and add it to the list of predicted obstacle trajectories that are evaluated in the obstacle prioritization scheme.

The task is to maintain a tight 8-agent formation, while a manually controlled UAV is flown through and generally tries to disrupt the formation, shown in Figure 3. All agents must keep the required distance to other agents in the distributed

scheme, while also avoiding the non-cooperative obstacle. The minimum distances in this experiment had an absolute minimum value of $0.33\,\mathrm{m}$ and a slightly lower over-all performance. Individual agents correctly select avoidance maneuvers that avoid the non-cooperative obstacle while minimally disrupting other agents. The simplified and non-dynamic prediction model of the non-cooperative agent cannot predict sudden turns or irregular movements, but despite this the safety-critical distance is still maintained and there was no collisions with any agent. The average solver time was $2.49\,\mathrm{ms}$, with a 0.12% non-convergence rate.

Fig. 1: Experiment set-up for first experiment. Top view, side view and visualization of predicted (position) trajectories in rviz. Agents are one-by-one tasked to move positions while the formation is maintained.
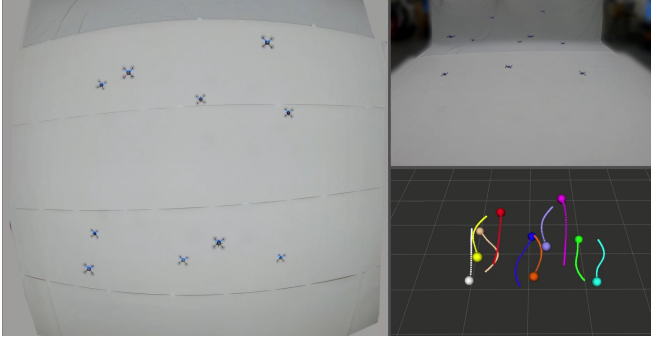

Fig. 2: Experiment set-up for second experiment. Two teams of five agents swap positions all at once.
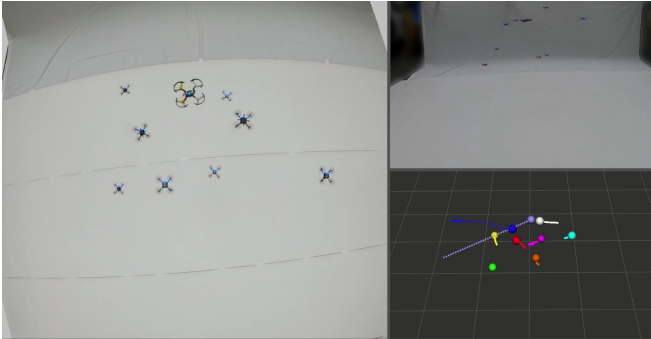

Fig. 3: Experiment set-up for third experiment. The larger UAV (purple) is manually controlled and act as the non-cooperative obstacle, here seen disrupting multiple agents.

III. CONCLUSIONS

In this paper we have presented a novel distributed collision avoidance scheme for Unmanned Aerial Vehicles (UAVs).
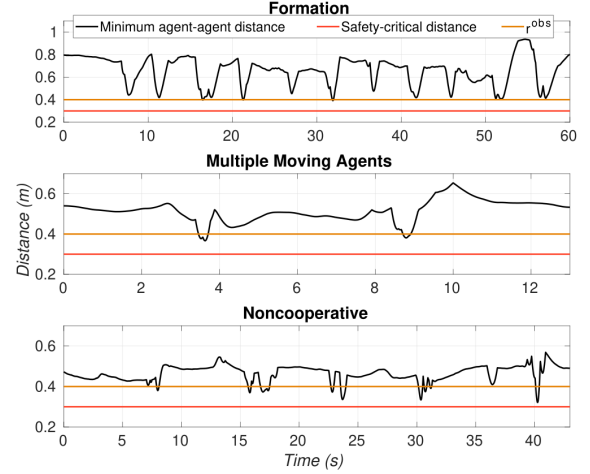

Fig. 4: Minimum agent-agent distances throughout the three experiments
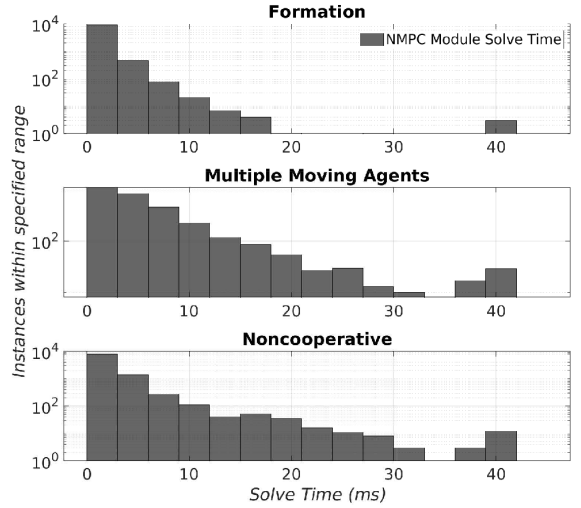

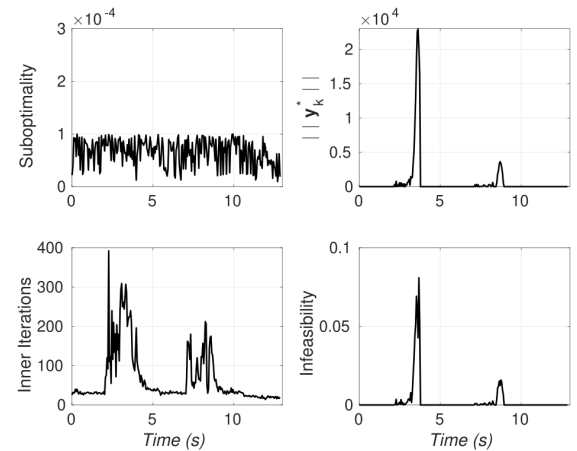Fig. 5: Histograms of solver times of the NMPC module

Fig. 6: Solver data from one of the agents. The suboptimality described by the norm of the fixed-point residual (top left), the norm of the Lagrange Multipliers (top right), Inner iterations (bottom left) and the constraint infeasibility (bottom right).

Through the presented results, we can conclude that the NMPC scheme based on the Optimization Engine (OpEn) can

provide real-time collision-free trajectories for all agents using a prediction horizon of two seconds. Also, the implemented augmented Lagrangian method has been demonstrated to be applicable to collision-avoidance constraints with tight run-time requirements. The obstacle prioritization correctly assigns which agents are to be considered as obstacles in time to safely avoid collisions, and allows a NMPC formulation with a low set number of constraints to fluidly deal with a large number of dynamic obstacles. It has also been shown that a constraint based obstacle avoidance scheme akin to the one we propose is much improved by good obstacle-trajectory information, as seen by the decreased performance when introducing the non-cooperative obstacle with a simplified predicted trajectory, a result similarly observed in [17]. NMPC lends itself to a distributed formulation since the motion intentions of all agents at every time instant is known based on the NMPC trajectories, but the question of how to accurately predict future obstacle positions of non-cooperative moving obstacles (and how to compensate if their trajectories are uncertain) is still a very interesting and open research question that we are currently working on.

REFERENCES

[1] S. S. Mansouri, C. Kanellakis, E. Fresk, D. Kominiak, and G. Niko-lakopoulos, "Cooperative coverage path planning for visual inspection," *Control Engineering Practice*, vol. 74, pp. 118–131, 2018.

[2] C. Nieto-Granda, J. G. Rogers III, and H. I. Christensen, "Coordination strategies for multi-robot exploration and mapping," *The International Journal of Robotics Research*, vol. 33, no. 4, pp. 519–533, 2014.

[3] Z. Beck, W. Teacy, N. Jennings, and A. Rogers, "Online planning for collaborative search and rescue by heterogeneous robot teams," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. Association of Computing Machinery, 2016.

[4] A. Budiyanto, A. Cahyadi, T. B. Adji, and O. Wahyunggoro, "UAV obstacle avoidance using potential field under dynamic environment," in *2015 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, 2015, pp. 187–192.

[5] A. Mondal, L. Behera, S. R. Sahoo, and A. Shukla, "A novel multi-agent formation control law with collision avoidance," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 3, pp. 558–568, 2017.

[6] G. M. Hoffmann and C. J. Tomlin, "Decentralized cooperative collision avoidance for acceleration constrained vehicles," in *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 4357–4363.

[7] T. Mylvaganam, M. Sassano, and A. Astolfi, "A differential game approach to multi-agent collision avoidance," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 4229–4235, 2017.

[8] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 477–483.

[9] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[10] C. K. Verginis and D. V. Dimarogonas, "Adaptive robot navigation with collision avoidance subject to 2nd-order uncertain dynamics," *arXiv preprint arXiv:2005.12599*, 2020.

[11] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Autonomous Robots*, vol. 39, no. 1, pp. 101–121, 2015.

[12] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

[13] A. Filotheou, A. Nikou, and D. V. Dimarogonas, "Decentralized control of uncertain multi-agent systems with connectivity maintenance and collision avoidance," in *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 8–13.

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

[15] B. Lindqvist, "Collision avoidance for multiple MAVs using fast centralized NMPC," in *International Federation of Control 2020*, 2020.

[16] L. Dai, Q. Cao, Y. Xia, and Y. Gao, "Distributed MPC for formation of multi-agent systems with collision avoidance and obstacle avoidance," *Journal of the Franklin Institute*, vol. 354, no. 4, pp. 2068–2085, 2017.

[17] H. Zhu and J. Alonso-Mora, "Chance-constrained collision avoidance for mavs in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 776–783, 2019.

[18] P. Sopasakis, E. Fresk, and P. Patrinos, "OpEn: Code generation for embedded nonconvex optimization," *arXiv preprint arXiv:2003.00292*, 2020.

[19] ——, "Optimization Engine," 2019. [Online]. Available: http://doc.optimization-engine.xyz/

[20] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 1939–1944.

[21] A. Sathya, P. Sopasakis, R. Van Parys, A. Themelis, G. Pipeleers, and P. Patrinos, "Embedded nonlinear model predictive control for obstacle avoidance using panoc," in *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 1523–1528.

[22] E. G. Birgin and J. M. Martínez, *Practical augmented Lagrangian methods for constrained optimization*. SIAM, 2014.

[23] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, "Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system," in *Robot operating system (ROS)*. Springer, 2017, pp. 3–39.

[24] E. Small, P. Sopasakis, E. Fresk, P. Patrinos, and G. Nikolakopoulos, "Aerial navigation in obstructed environments with embedded nonlinear model predictive control," in *2019 18th European Control Conference (ECC)*, 2019, pp. 3556–3563.

[25] Bitcraze AB, "Crazyflie 2.1," 2020. [Online]. Available: https://www.bitcraze.io/crazyflie-2-1/

[26] W. Hönig and N. Ayanian, *Flying Multiple UAVs Using ROS*. Springer International Publishing, 2017, pp. 83–118.

[27] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3. Kobe, Japan, 2009, p. 5.

[28] B. Lindqvist, S. S. Mansouri, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Nonlinear MPC for collision avoidance and control of UAVs with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6001–6008, 2020.