Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control

Kamel, Mina; Alonso-Mora, Javier; Siegwart, Roland; Nieto, Juan

DOI
10.1109/IROS.2017.8202163

Publication date
2017

Document Version
Accepted author manuscript

Published in
Proceedings 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

Robust Collision Avoidance for Multiple Micro Aerial Vehicles Using Nonlinear Model Predictive Control

Mina Kamel*, Javier Alonso-Mora†, Roland Siegwart*, and Juan Nieto*
*Autonomous Systems Lab, ETH Zurich
†Cognitive Robotics, Delft University of Technology

***Abstract*— When several Multirotor Micro Aerial Vehicles (MAVs) share the same airspace, reliable and robust collision avoidance is required. In this paper we address the problem of multi-MAV reactive collision avoidance. We employ a model-based controller to simultaneously track a reference trajectory and avoid collisions. Moreover, to achieve a higher degree of robustness, our method also accounts for the uncertainty of the state estimator and of the position and velocity of the other agents. The proposed approach is decentralized, does not require a collision-free reference trajectory and accounts for the full MAV dynamics. We validated our approach in simulation and experimentally with two MAV.**

I. INTRODUCTION

As the miniaturization technology advances, low cost and reliable MAVs are becoming available on the market with powerful on-board computation power. Many applications can benefit from the presence of low cost systems, such as inspection and exploration [1], [2], surveillance [3], mapping [4] and aerial videography [5]. However, MAVs are limited to short flight times due to battery limitation and size constraints. Due to these limitations, creating a team of MAVs that can safely share the airspace to execute a specific mission would be beneficial for time-critical missions such as search and rescue operations [6] and would also widen the range of applications where MAVs can be used.

A crucial problem when multiple MAVs share the same airspace is the risk of mid-air collision. Because of this, a robust method to avoid multi-MAVs collisions is necessary. Typically, this problem is solved by planning a collision-free trajectory for each agent in a centralized manner. However, this binds the MAVs to the pre-planned trajectory and limits the adaptivity of the team during the mission: any change in the task would require trajectory re-planning for the whole team.

In this work we present a unified framework to achieve reference trajectory tracking and multi-agent reactive collision avoidance. The proposed approach exploits the full MAV dynamics and takes into account the limitations of the physical platform. In this way we fully exploit the MAV capabilities and achieve agile and natural avoidance maneuvers compared to classic approaches, where planning is decoupled from trajectory tracking control. To this end, we formulate the control problem as a constrained optimization problem that we solve in a receding horizon fashion. The cost function of the optimization problem includes a potential field-like term that penalizes collisions between agents. While potential field methods do not provide any guarantee and are sensitive to tuning parameters, we introduce additional tight hard constraints to guarantee that no mid-air collisions will occur. The proposed method assumes that each agent knows the position and velocity of all other agents. Additionally, to increase the avoidance robustness, we use the state estimator uncertainty to shape the collision term in the cost function and the optimization constraints.

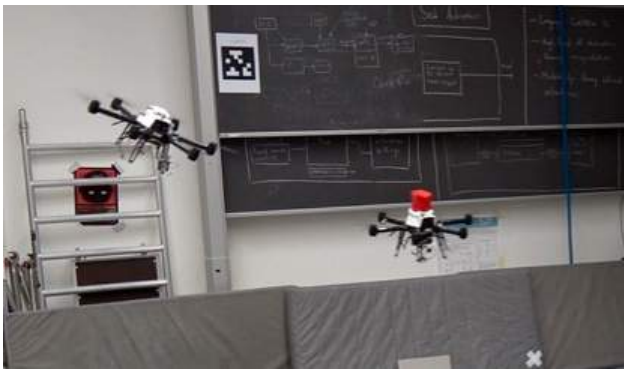


Fig. 1: An instance of the experimental evaluation of the proposed method for multi-agent collision avoidance.

The contribution of this paper can be summarized as follows:

i A unified framework for multi-agent control and collision avoidance.
ii The incorporation of state estimator uncertainty and communication delay for robust collision avoidance.

This paper is organized as follows. In Section II we present an overview of existing methods for multi-agent collision avoidance. In Section III we briefly present the MAV model that will be considered in the controller formulation. In Section IV we present the controller and discuss the state estimator uncertainty propagation. Finally, in Section V we present simulation and experimental results of the proposed approach.

II. RELATED WORK

Many researchers have demonstrated successful trajectory generation and navigation on MAVs in controlled environment where obstacles are static, using external motion capture system [7] or using on-board sensing [8]. Sampling based planning techniques can be used to generate global collision-free trajectories for a single agent, taking into account the agent dynamics [9] and static and dynamics obstacles in the environment.

One way to generate collision-free trajectories for a team of robots is to solve a mixed integer quadratic problem in a centralized fashion as shown in [10]. A similar approach was presented in [11] where sequential quadratic programming techniques were employed to generate collision-free trajectories for a team of MAVs. The aforementioned methods lack real-time performance and do not consider unforeseen changes in the environment.

Global collision-free trajectory generation methods limit the versatility of the team of robots. In real missions, where multiple agents are required to cooperate, the task assigned to each agent might change according to the current situation, and real-time reactive methods for local trajectory planning become crucial.

One of the earliest works to achieve reactive collision avoidance for a team of flying robots using a Nonlinear Model Predictive Control (NMPC) framework is the work presented in [12]. A decentralized NMPC was employed to control multiple helicopters in a complex environment with an artificial potential field for reactive collision avoidance. This approach does not provide any guarantees and was evaluated only in simulation.

In [13] the authors present various algorithms based on the Velocity Obstacles (VO) concept to select collision-free trajectories from a set of candidate trajectories. The method was experimentally evaluated on four MAVs flying in close proximity and including human. However, the MAV dynamics were not fully modeled, and decoupling trajectory generation from control has various limitations as shown in the experimental section of [13].

Among the attempts to unify trajectory optimization and control is the work presented in [14]. The robot control problem is formulated as a finite horizon optimal control problem and an unconstrained optimization is performed at every time step to generate time-varying feedback gains and feed-forward control inputs simultaneously. The approach has been successfully applied on MAVs and a ball balancing robot. Similar to our work, [5] also employed a NMPC framework for controlling a MAVs while avoiding collisions, but limited to a single MAVs. We employ a more general dynamical model for the MAVs.

In this work, we unify the trajectory tracking and collision avoidance into a single optimization problem in a decentralized manner. In this way, trajectories generated from a global planner can be sent directly to the trajectory tracking controller without modifications, leaving the local avoidance task to the tracking controller.

III. MODEL

In this section we present the MAV model employed in the controller formulation. We first introduce the full vehicle model and explain the forces and moments acting on the system. Next, we will briefly discuss the closed-loop attitude model employed in the trajectory tracking controller.

1) System model: We define the world fixed inertial frame I and the body fixed frame B attached to the MAV in the Center of Gravity (CoG) as shown in Figure 2. The vehicle

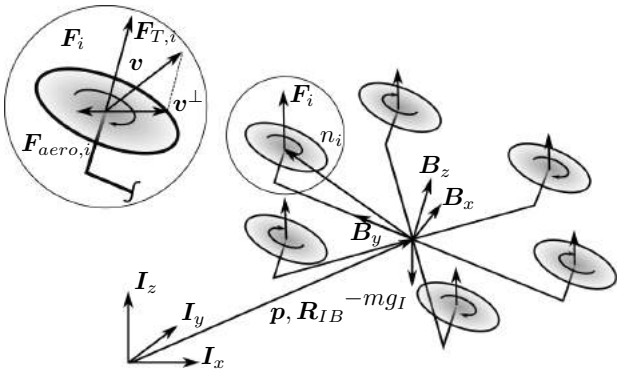


Fig. 2: A schematic of a MAV showing forces and torques acting on the MAV, and aerodynamic forces acting on a single rotor. Inertial and CoG frames are also shown.

configuration is described by the position of the CoG in the inertial frame $\boldsymbol{p} \in \mathbb{R}^3$, the vehicle velocity in the inertial frame $\boldsymbol{v}$, the vehicle orientation $\boldsymbol{R}_{IB} \in SO(3)$ which is parameterized by Euler angles and the body angular rate $\boldsymbol{\omega}$.

The main forces acting on the vehicle are generated from the propellers. Each propeller generates thrust proportional to the square of the propeller rotation speed $n_i$and angular moment due to the drag force. The generated thrust $\boldsymbol{F}_{T,i}$ and moment $\boldsymbol{M}_i$ from the $i-th$ propeller is given by:

$$\boldsymbol{F}_{T,i} = k_n n_i^2 \boldsymbol{e}_z, \tag{1a}$$

$$\boldsymbol{M}_i = (-1)^{i-1} k_m \boldsymbol{F}_{T,i}, \tag{1b}$$

where k_n and k_m are positive constants and $\boldsymbol{e}_z$ is a unit vector in z direction. Moreover, we consider two important effects that appear in the case of dynamic maneuvers. These effects are the blade flapping and induced drag. The importance of these effects stems from the fact that they introduce additional forces in the $x-y$ rotor plane, adding some damping to the MAV velocity as shown in [15]. It is possible to combine these effects as shown in [16], [17] into one lumped drag coefficient k_D.

This leads to the aerodynamic force $\boldsymbol{F}_{aero,i}$:

$$\boldsymbol{F}_{aero,i} = f_{T,i} \boldsymbol{K}_{drag} \boldsymbol{R}_{IB}^T \boldsymbol{v} \tag{2}$$

where $\boldsymbol{K}_{drag} = diag(k_D, k_D, 0)$ and $f_{T,i}$ is the z component of the $i-th$ thrust force.

The motion of the vehicle can be described by the follow-

ing equations:

$$\dot{\boldsymbol{p}} = \boldsymbol{v}, \tag{3a}$$

$$\dot{\boldsymbol{v}} = \frac{1}{m}\left(\boldsymbol{R}_{IB}\sum_{i=0}^{N_r}\boldsymbol{F}_{T,i} - \boldsymbol{R}_{IB}\sum_{i=0}^{N_r}\boldsymbol{F}_{aero,i} + \boldsymbol{F}_{ext}\right) + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}, \tag{3b}$$

$$\dot{\boldsymbol{R}}_{IB} = \boldsymbol{R}_{IB}\lfloor\boldsymbol{\omega}\times\rfloor, \tag{3c}$$

$$\boldsymbol{J}\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega}\times\boldsymbol{J}\boldsymbol{\omega} + \boldsymbol{\mathcal{A}}\begin{bmatrix} n_1^2 \\ \vdots \\ n_{N_r}^2 \end{bmatrix}, \tag{3d}$$

where m is the mass of the MAV and $\boldsymbol{F}_{ext}$ is the external forces acting on the vehicle (i.e wind). $\lfloor\boldsymbol{\omega}\times\rfloor$ is the skew symmetric matrix of the angular velocity expressed in body frame. $\boldsymbol{J}$ is the inertia matrix, $\boldsymbol{\omega}$ is the angular velocity, $\mathcal{A}$ is the allocation matrix and N_r is the number of propellers.

2) Attitude model: We follow a cascaded approach as described in [18] and assume that the vehicle attitude is controlled by an attitude controller. For completeness we quickly summarize their findings in the following paragraph.

To achieve accurate trajectory tracking, it is crucial for the high level controller to consider the inner loop system dynamics. Therefore, it is necessary to consider a simple model of the attitude closed-loop response. These dynamics can either be calculated by simplifying the closed loop dynamic equations (if the controller is known) or by a simple system identification procedure in case of an unknown attitude controller (on commercial platforms for instance). In this work we used the system identification approach to identify a first order closed-loop attitude response.

The inner-loop attitude dynamics are then expressed as follows:

$$\dot{\phi} = \frac{1}{\tau_{\,\phi}}\left(k_\phi\phi_{cmd} - \phi\right), \tag{4a}$$

$$\dot{\theta} = \frac{1}{\tau_{\,\theta}}\left(k_\theta\theta_{cmd} - \theta\right), \tag{4b}$$

$$\dot{\psi} = \dot{\psi}_{cmd}, \tag{4c}$$

where k_ϕ, k_θ and τ_ϕ, τ_θ are the gains and time constant of roll and pitch angles respectively. ϕ_{cmd} and θ_{cmd} are the commanded roll and pitch angles and $\dot{\psi}_{cmd}$ is the commanded angular velocity of the vehicle heading.

The aforementioned model will be employed in the subsequent trajectory tracking controller to account for the attitude inner-loop dynamics. Note that the vehicle heading angular rate $\dot{\psi}$ is assumed to track the command instantaneously. This assumption is reasonable as the MAV heading angle has no effect on the MAV position.

IV. CONTROLLER FORMULATION

In this section we present the unified trajectory tracking and multi-agent collision avoidance NMPC controller. First, we will present the Optimal Control Problem (OCP). Afterwards we will discuss the cost function choice and the state estimator uncertainty propagation to achieve robust collision avoidance. Next, we will present the optimization constraints and finally we will discuss the approach adopted to solve the OCP in real-time on-board of the MAV.

A. Optimal Control Problem

To formulate the OCP, we first define the system state vector $\boldsymbol{x}$ and control input $\boldsymbol{u}$ as follows:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}^T & \boldsymbol{v}^T & \phi & \theta & \psi \end{bmatrix}^T \tag{5}$$

$$\boldsymbol{u} = \begin{bmatrix} \phi_{cmd} & \theta_{cmd} & T_{cmd} \end{bmatrix}^T \tag{6}$$

Every time step, we solve the following OCP online:

$$\begin{aligned} \min_{\boldsymbol{U},\boldsymbol{X}} \; & \int_{t=0}^{T}\left\{J_{\boldsymbol{x}}\left(\boldsymbol{x}(t),\boldsymbol{x}_{ref}(t)\right) + J_{\boldsymbol{u}}\left(\boldsymbol{u}(t),\boldsymbol{u}_{ref}(t)\right) + J_c\left(\boldsymbol{x}(t)\right)\right\}dt \\ & + J_T\left(\boldsymbol{x}(T)\right) \\ \text{subject to} \quad & \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}); \\ & \boldsymbol{u}(t) \in \mathbb{U} \\ & \boldsymbol{G}(\boldsymbol{x}(t)) \leq 0 \\ & \boldsymbol{x}(0) = \boldsymbol{x}\left(t_0\right). \end{aligned} \tag{7}$$

where $\boldsymbol{f}$ is composed of Equations (3a), (3b) and (4). $J_{\boldsymbol{x}}$ is the cost function for tracking the reference trajectory $\boldsymbol{x}_{ref}$, $J_{\boldsymbol{u}}$ is the control input penalty, J_c is the collision cost and J_T is the terminal cost function. $\boldsymbol{G}$ is a function that represents the state constraint, and $\mathbb{U}$ is the set of admissible control inputs. In the following, we discuss the details of the aforementioned OCP and discuss a method to efficiently solve it in real-time.

B. Cost Function

We now describe the components of the cost function presented in Equation (7). The first term $J_{\boldsymbol{x}}\left(\boldsymbol{x}(t),\boldsymbol{x}_{ref}(t)\right)$ penalizes the deviation of the predicted state $\boldsymbol{x}$ from the desired state vector $\boldsymbol{x}_{ref}$ in a quadratic sense as shown below:

$$J_{\boldsymbol{x}}\left(\boldsymbol{x}(t),\boldsymbol{x}_{ref}(t)\right) = \left\|\boldsymbol{x}(t) - \boldsymbol{x}_{ref}(t)\right\|^2_{\boldsymbol{Q}_x} \tag{8}$$

where $\boldsymbol{Q}_x \succeq 0$ is a tuning parameter. The state reference $\boldsymbol{x}_{ref}$ is obtained from the desired trajectory. The second term in the cost function is related to the penalty on the control input as shown below:

$$J_{\boldsymbol{u}}\left(\boldsymbol{u}(t),\boldsymbol{u}_{ref}(t)\right) = \left\|\boldsymbol{u}(t) - \boldsymbol{u}_{ref}(t)\right\|^2_{\boldsymbol{R}_u} \tag{9}$$

where $\boldsymbol{R}_u \succeq 0$ is a tuning parameter. The control input reference $\boldsymbol{u}_{ref}$ is chosen to achieve better tracking performance based on desired trajectory acceleration as described in [19].

The collision cost $J_c\left(\boldsymbol{x}(t)\right)$ to avoid collisions with other N_{agents} is based on the logistic function, and the main motivation behind this choice is to achieve a smooth and bounded collision cost function. Figure 3 shows the cost

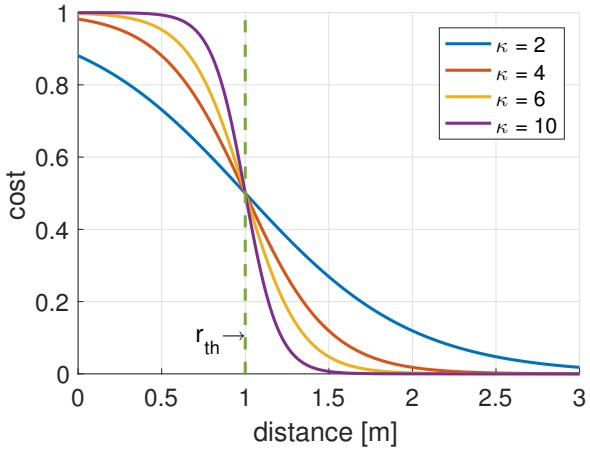


Fig. 3: Logistic function based potential field for different smoothness parameter κ.

function for different κ parameters. The collision cost is given by:

$$J_c\left(\boldsymbol{x}(t)\right) = \sum_{j=1}^{N_{agents}} \frac{Q_{c,j}}{1+\exp\kappa_j\left(d_j(t) - r_{th,j}(t)\right)} \quad (10)$$
$$\text{for } j = 1, \ldots, N_{agents}$$

where $d_j(t)$ is the Euclidean distance to the $j{-}th$ agent given by $d_j(t) = \|\boldsymbol{p}(t) - \boldsymbol{p}_j(t)\|_2$, $Q_{c,j} > 0$ is a tuning parameter, $\kappa_j > 0$ is a parameter that defines the smoothness of the cost function and $r_{th,j}(t)$ is a threshold distance between the agents where the collision cost is $Q_{c,j}/2$.

C. Constraints

The first constraint in (7) guarantees that the state evolution respects the MAV dynamics. To achieve offset-free tracking we must compensate for external disturbances and modeling errors to achieve offset-free tracking. We employ a model-based filter to estimate external disturbances $\boldsymbol{F}_{ext}$ as described in details in [19]. The second constraint addresses limitations on the control input:

$$\mathbb{U} = \left\{ \boldsymbol{u} \in \mathbb{R}^3 | \begin{bmatrix} \phi_{min} \\ \theta_{min} \\ T_{cmd,min} \end{bmatrix} \leq \boldsymbol{u} \leq \begin{bmatrix} \phi_{max} \\ \theta_{max} \\ T_{cmd,max} \end{bmatrix} \right\}. \quad (11)$$

The third constraint guarantees collision avoidance by setting tight hard constraints on the distance between two agents. The $j-th$ row of the $\boldsymbol{G}$ matrix represents the collision constraints with the $j-th$ agent. This is given by:

$$\boldsymbol{G}_j(\boldsymbol{x}) = -\|\boldsymbol{p}(t) - \boldsymbol{p}_j(t)\|_2^2 + r_{min,j}^2(t). \quad (12)$$

where $\boldsymbol{p}_j(t)$ is the position of the $j-th$ agent at time t. These are non-convex constraints and $\boldsymbol{G}$ is continuous and smooth. $r_{min,j}$ is chosen to always be strictly less than $r_{th,j}$ to guarantee that the hard constraints are activated only if the potential field in Equation (10) is not able to maintain $r_{min,j}$ distance to the $j{-}th$ agent. For simplicity we consider spherical hard constraints, but this can be replaced with more complex geometry.

Finally, the last constraint in the optimization problem is to fix the initial state $\boldsymbol{x}(0)$ to the current estimated state $\boldsymbol{x}(t_0)$.

D. Motion Prediction of Agents

Since the approach presented in this work is based on Model Predictive Control, it is beneficial to employ a simple model for the other agents and use it to predict their future behavior. In this work we assume that the agents communicate their current position and velocity through a common network, and employ a constant velocity model, which could be replaced by a more sophisticated model. Given the current position and velocity of the $j-th$ agent $\boldsymbol{p}_j(t_0), \boldsymbol{v}_j(t_0)$ we predict the future positions of the $j-th$ agent along the prediction horizon as follows:

$$\boldsymbol{p}_j(t) = \boldsymbol{p}_j(t_0) + \boldsymbol{v}_j(t_0)\left(t - t_0 + \delta\right). \quad (13)$$

where δ is the communication delay that we compensate for to achieve better prediction. δ is calculated based on the difference between the timestamp on the message and the arrival time. This is possible thanks to a clock synchronization between the agents and a time server. The communication delay compensation can be omitted if there is no clock synchronization between agents. Additionally, to reduce the noise sensitivity, i.e. we consider the velocity to be zero if it is below a certain threshold v_{th}.

E. Uncertainty Propagation

To achieve a higher level of robustness, we account for the uncertainty in the state estimator and the uncertainty of the other agents. We propagate the estimated state uncertainty to calculate the minimum allowed distance $r_{min,j}(t)$ to the $j-th$ agent and the threshold distance $r_{th,j}(t)$. In other words, if the state is highly uncertain, we should be more conservative on allowing agents to get closer to each other by increasing $r_{min,j}$ and $r_{th,j}$ at time t along the prediction horizon. The uncertainty of the $j{-}th$ agent's position is propagated using the model described in Equation (13), while the self-uncertainty can be propagated with higher accuracy employing the system model described in Equations (3a), (3b) and (4). In many previous works, the uncertainty propagation is typically performed using the unscented transformation when the system is nonlinear. In our case, given that we need real-time performance, we choose to perform uncertainty propagation based on an Extended Kalman Filter (EKF). Given the current predicted state $\boldsymbol{x}(t_0)$ with covariance $\boldsymbol{\Sigma}(t_0)$, we propagate the uncertainty by solving the following differential equation:

$$\dot{\boldsymbol{\Sigma}}(t) = \boldsymbol{F}(t)\boldsymbol{\Sigma}(t)\boldsymbol{F}(t)^T \quad (14)$$

with boundary condition $\boldsymbol{\Sigma}(0) = \boldsymbol{\Sigma}(t_0)$. $\boldsymbol{F}(t)$ is the state transition Jacobian matrix. Using Equation (14) we compute the $j-th$ agent's uncertainty and the self-uncertainty at time t, namely $\boldsymbol{\Sigma}_j(t)$ and $\boldsymbol{\Sigma}(t)$. These values are employed to calculate $r_{min,j}(t)$ and $r_{th,j}(t)$. We use the maximum eigenvalue to reduce the problem of computing the distance between two ellipsoids, which is more complex and time consuming, to the computation of the distance between two spheres. Therefore, $r_{min,j}(t)$ and $r_{th,j}(t)$ are calculated

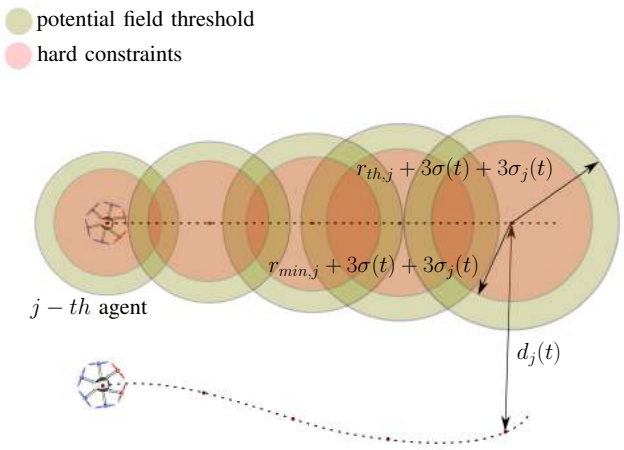


Fig. 4: The concept of robust collision avoidance, the minimum acceptable distance between two agents is increased along the prediction horizon based on the state uncertainty propagation. Tighter hard constraints will be activated only if the potential field fails to maintain the minimum acceptable distance $r_{min,j}$. Note that the plotted trajectories are predicted trajectories obtained from the solution of the OCP (7).

according to the following equations:

$$\begin{aligned} r_{min,j}(t) &= r_{min} + 3\sigma(t) + 3\sigma_j(t), \\ r_{th,j}(t) &= r_{th} + 3\sigma(t) + 3\sigma_j(t). \end{aligned} \tag{15}$$

where σ is the square root of the maximum eigenvalue of the self-uncertainty $\mathbf{\Sigma}$ and σ_j is the square root of maximum eigenvalue of the $j-th$ agent's uncertainty $\mathbf{\Sigma}_j$. r_{min} and r_{th} are constant parameters. Approximating the uncertainty ellipsoid by the enclosing sphere makes the bounds more conservative, especially if the uncertainty is disproportionately large only in a particular direction. Figure 4 illustrates the concept for two agents.

F. Implementation

A multiple shooting technique [20] is employed to solve the OCP (7). The system dynamics and constraints are discretized over a coarse discrete time grid $t_0, \dots, t_N$ within the time interval $[t_k, t_{k+1}]$. For each interval, a Boundary Value Problem (BVP) is solved, where additional continuity constrains are imposed. An implicit *RK* integrator of order 4 is employed to forward simulate the system dynamics along the interval. The OCP can be expressed as a Nonlinear Program (NLP) which we solve using Sequential Quadratic Programming (SQP). An active set or interior point method can be used to solve the associated Quadratic Program (QP). For the controller to behave as a local planner and to guarantee problem feasibility, a long prediction horizon T is necessary. To achieve this without significantly increasing the computation effort, the time step of the grid over which the system dynamics is discretized is chosen to be larger than the controller rate. Smooth predictions are obtained thanks to the implicit Runge-Kutta of order 4 integrator employed and because the system dynamics is not represented by stiff differential equations.

V. EVALUATION

In this section we evaluate the proposed approach experimentally and in simulation. The proposed controller has been implemented in C++ and integrated into Robot Operating System (ROS). The ACADO toolkit [21] is employed to generate a fast C solver for the OCP.

First we present two experimental evaluations of the method with two MAVs. Then we show two simulation studies in a high fidelity MAVs simulator, RotorS [22] where six MAVs are commanded to swap positions simultaneously.

The proposed method can handle priorities simply by changing the connectivity graph between agents. The highest priority MAV will not perform any avoidance maneuver and therefore doesn't need to have access to other agents state. A lower priority agent will perform avoidance maneuver only to avoid higher priority agents.

A. Experimental Results

We experimentally evaluated the proposed approach on two AscTec NEO hexacopters. In a first experiment, one MAV is commanded to hover in position, while a second MAV with high priority is commanded manually to fly in the vicinity to the first MAV. In a second experiment, crossing reference trajectories are planned for both MAVs and executed. The purpose of these experiments is to show how the proposed method exploits the system dynamics and abruptly responds to changes in the other agent behavior in a robust manner.

1) Experimental Setup: The AscTec NEO hexacopter is equipped with an Intel i7 2.8 GHz 8 GHz RAM computer running ROS and an onboard flight controller and tuned attitude controller. The on-board computer communicates with the flight controller over a serial port and exchanges information at 100 Hz. An external motion capture system (Vicon) is used to measure each MAV pose. This measurement is fused with the onboard Inertial Measurement Unit (IMU) to achieve accurate state estimation using the multi-sensor fusion framework [23]. The estimated position and velocity of each agent is shared over the network. The controller is running onboard of each MAV at 100 Hz while the prediction horizon of the controller is chosen to be 2 seconds.

2) Experiments: Figure 5 shows a sequence of images taken 1 second apart during the experiment [1].

Figure 6a shows the distance between the two agents over time. The distance is almost always maintained above r_{th} except for moments of aggressive maneuvers, however the hard constraint was never activated as the distance was always above r_{min} all the time.

In the second experiment, intersecting reference trajectories with maximum velocity of 2 m/s are planned and sent as reference trajectories to the respective MAV. Figure 6b shows the distance between the two MAVs over time. The hard constraint was never active since the distance was always above r_{min}.

[1] video available on `goo.gl/RWRhmJ`

B. Simulation Results

In this simulation study we evaluate our method on six MAVs commanded to exchange their initial positions. The reference trajectory for each agent is not collision-free. Figure 7a shows the evolution of the MAV trajectories when all agents are sharing position and velocity information in a fully connected graph. In this case, avoidance maneuvers are reciprocal. The average computation time during this simulation is 1.0 ms, while worst case computation time is around 4.0 ms on an Intel i7 2.8 GHz CPU.

In another simulation study depicted in Figure 7b, we assigned a hierarchical avoidance scheme, where the first MAV (blue sphere) has top priority to follow the reference trajectory, while the second MAV (red sphere) is avoiding only the first MAV. The third MAV is avoiding the first two, etc... .

In this case, assigning a priority scheme makes the problem simpler to solve and with better position swapping trajectories as shown in Figure 7.

VI. CONCLUSIONS

In this paper we presented a multi-MAVs collision avoidance strategy based on Nonlinear Model Predictive Control. The approach accounts for state estimator uncertainty by propagating the uncertainty along the prediction horizon to increase the minimum acceptable distance between agents, providing robust collision avoidance. Tight hard constraints on the distance between agents guarantee collision-free navigation if the prediction horizon is sufficiently long. Moreover, by changing the connectivity graph, it is possible to assign priority to certain agents to follow their reference trajectories. The approach has been evaluated in simulation with 6 simulated MAVs and in real experiments with two MAVs. Our experiments showed that this collision avoidance approach results into agile and dynamic avoidance maneuvers while maintaining system stability at reasonable computational cost.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644128 and from the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 15.0044.

REFERENCES

[1] K. Steich, M. Kamel, P. Beardsleys, M. K. Obrist, R. Siegwart, and T. Lachat, "Tree cavity inspection using aerial robots," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.

[2] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon "next-best-view" planner for 3d exploration," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1462–1468.

[3] A. Girard, A. Howell, and J. Hedrick, "Border patrol and surveillance missions using multiple unmanned air vehicles," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, 2004.

[4] P. Oettershagen, T. J. Stastny, T. A. Mantel, A. S. Melzer, K. Rudin, G. Agamennoni, K. Alexis, R. Siegwart, "Long-endurance sensing and mapping using a hand-launchable solar-powered uav," in *Field and Service Robotics, 10th Conference on*, June 2015, (accepted).

[5] T. Nägeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-time motion planning for aerial videography with real-time with dynamic obstacle avoidance and viewpoint optimization," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1696–1703, July 2017.

[6] P. Rudol and P. Doherty, "Human body detection and geolocalization for uav search and rescue missions using color and thermal imagery," in *Aerospace Conference, 2008 IEEE*, 2008, pp. 1–8.

[7] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *IEEE Robotics & Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[8] M. Burri, H. Oleynikova, M. W. Achtelik, and R. Siegwart, "Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 1872–1878.

[9] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[10] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, 2013.

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

[12] D. H. Shim, H. J. Kim, and S. Sastry, "Decentralized nonlinear model predictive control of multiple flying robots," in *Decision and control, 2003. Proceedings. 42nd IEEE conference on*, vol. 4. IEEE, 2003, pp. 3621–3626.

[13] 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.

[14] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1398–1404.

[15] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, Sept 2012.

[16] S. Omari, M. D. Hua, G. Ducard, and T. Hamel, "Nonlinear control of vtol uavs incorporating flapping dynamics," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 2419–2425.

[17] M. Burri, J. Nikolic, H. Oleynikova, M. W. Achtelik, and R. Siegwart, "Maximum likelihood parameter identification for mavs," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 4297–4303.

[18] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based mav navigation in unknown and unstructured environments," in *Robotics and automation (ICRA), 2010 IEEE international conference on*. IEEE, 2010, pp. 21–28.

[19] 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 International Publishing, 2017, pp. 3–39.

[20] C. Kirches, *The Direct Multiple Shooting Method for Optimal Control*. Wiesbaden: Vieweg+Teubner Verlag, 2011, pp. 13–29. [Online]. Available: http://dx.doi.org/10.1007/978-3-8348-8202-8_2

[21] B. Houska, H. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[22] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Cham: Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-26054-9_23

[23] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3923–3929.

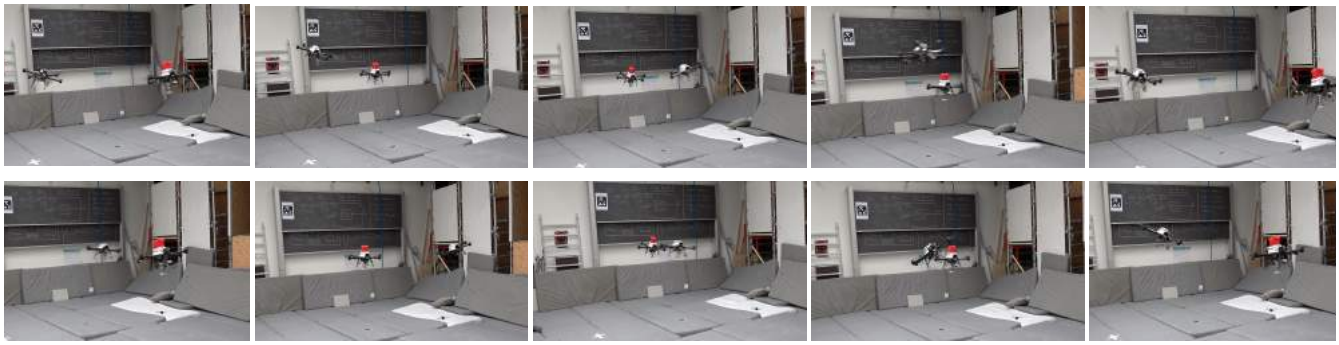
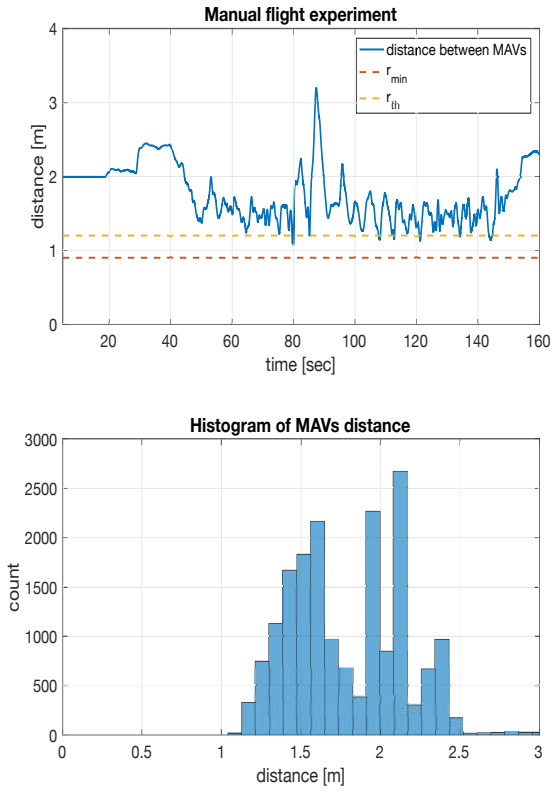
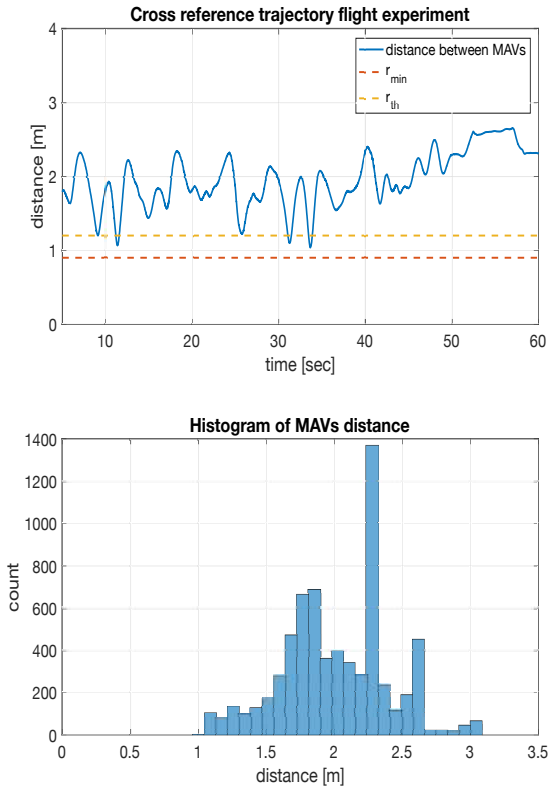


Fig. 5: A sequence of images during the cross trajectories experiments. The two MAVs are commanded to follow a non collision-free trajectories with priority assigned to the MAV with the red hat. Images are taken 1 second apart starting from top left.

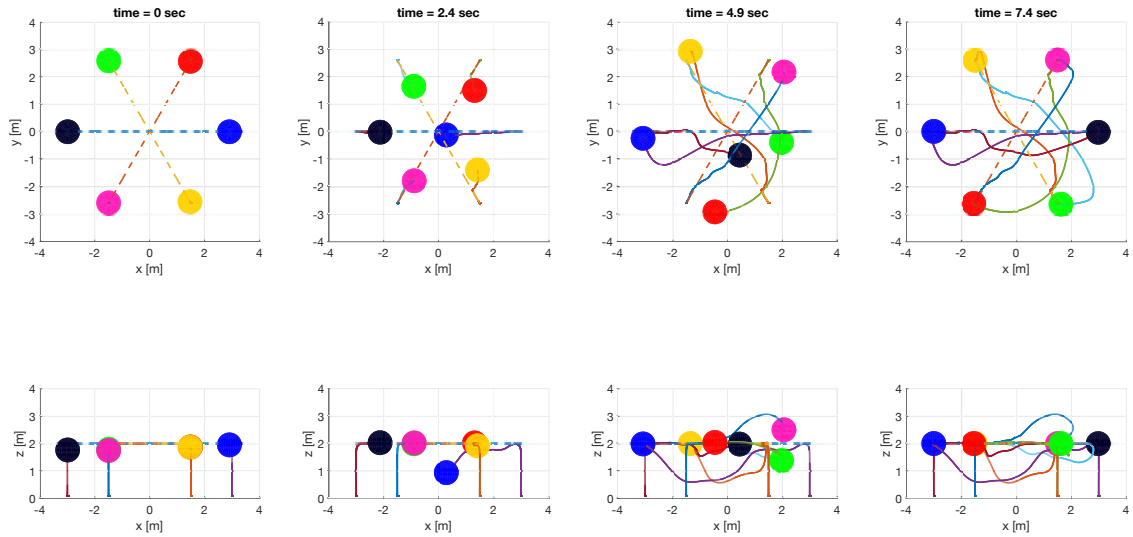


(a) The upper plot shows the distance between two MAVs over time during the manual flight experiment. One MAV is commanded to hover in position while the other one is manually commanded to approach it with a priority assigned to the manually commanded MAV. r_{th} is set to 1.2 m while the hard constraint on the distance is set to $r_{min} = 0.9$ m. The lower plot shows the histogram of the distance.

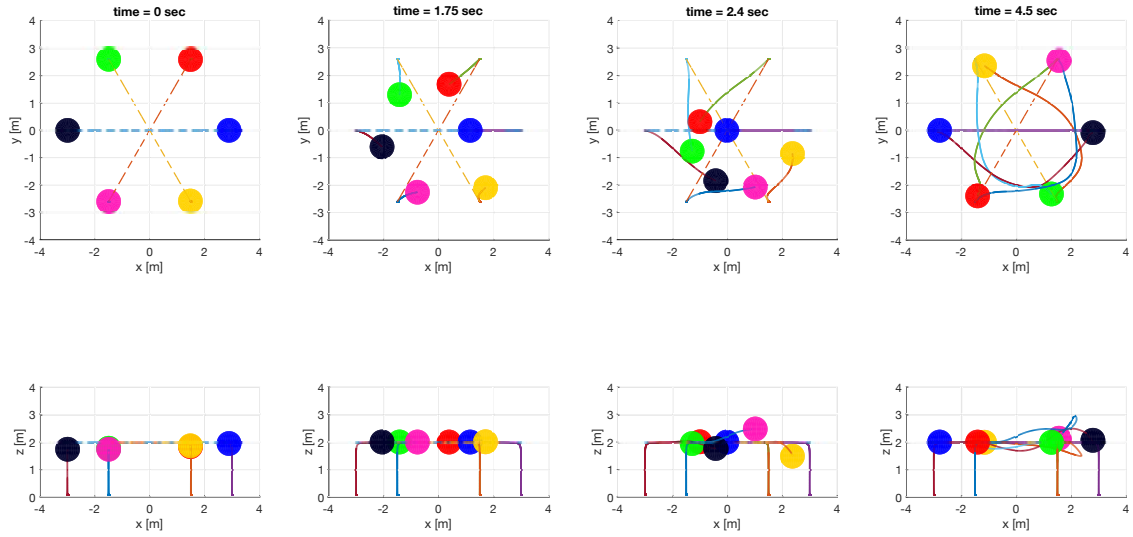


(b) The upper plot shows the distance between two MAVs over time during the crossing trajectories experiment. r_{th} is set to 1.2 m while the hard constraint on the distance is set to $r_{min} = 0.9$ m. The lower plot shows the histogram of the distance.

Fig. 6: Experimental results with two MAVs



(a) Trajectories of 6 MAVs during a position swapping simulation. The MAVs are represented by spheres, while dotted lines represent the reference trajectory provided to the controller. Solid lines represent the actual trajectory executed by each agent.



(b) Trajectories of 6 MAVs during a position swapping simulation with priority given to the first MAV (blue sphere). Dotted lines represent the reference trajectory provided to the controller. Solid lines represent the actual trajectory executed by each agent.

Fig. 7: Simulation results of 6 MAVs exchanging positions