Implementation and validation of an event-based real-time nonlinear model predictive control framework with ROS interface for single and multi-robot systems.

DENTLER, J., KANNAN, S., OLIVARES-MENDEZ, M.A. and VOOS, H.

2017

Implementation and Validation of an Event-Based Real-Time Nonlinear Model Predictive Control Framework with ROS Interface for Single and Multi-robot Systems*

Jan Dentler, Somasundar Kannan, Miguel A. Olivares-Mendez, Holger Voos[1]

***Abstract*— This paper presents the implementation and experimental validation of a central control framework. The presented framework addresses the need for a controller, which provides high performance combined with a low-computational load while being on-line adaptable to changes in the control scenario. Examples for such scenarios are cooperative control, task-based control and fault-tolerant control, where the system's topology, dynamics, objectives and constraints are changing. The framework combines a fast Nonlinear Model Predictive Control (*NMPC*), a communication interface with the Robot Operating System (*ROS*) [1] as well as a modularization that allows an event-based change of the *NMPC* scenario. To experimentally validate performance and event-based adaptability of the framework, this paper is using a cooperative control scenario of Unmanned Aerial Vehicles (*UAV*s). The source code of the proposed framework is available under [2].**

I. INTRODUCTION

The cooperative use of mobile robots in transportation, surveillance, maintenance, etc. has increased over the last decades. The combination of multiple specialized robots allows executing versatile tasks in a highly efficient way and can furthermore provide safety redundancy. To fully exploit this versatility, the control of the system has to adapt to the specified task, utilized robots and environment. The challenging nature of such dynamically changing cooperative control tasks has motivated the development of the here presented control framework.

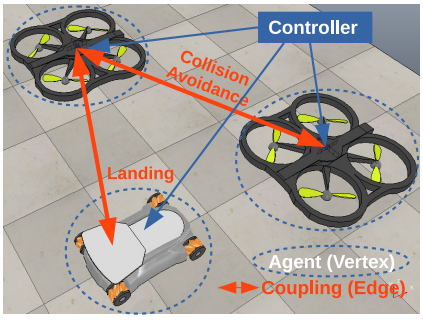


Fig. 1: Centrally controlled cooperative control scenario[3]

*This work was supported by FNR "Fonds national de la Recherche" (Luxembourg) through AFR "Aides à la Formation-Recherche" Ph.D. grant scheme No. 9312118.

[1]Jan Dentler, Dr. Somasundar Kannan, Dr. Miguel A. Olivares-Mendez and Prof. Dr.-Ing Holger Voos are with Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, L-1359 Luxembourg, Luxembourg jan.dentler@uni.lu, somasundar.kannan@uni.lu, miguel.olivaresmendez@uni.lu, holger.voos@uni.lu

For large-scale multi-robot scenarios typically distributed control solutions are applied due to computational limits. However, the adaptation of such decentralized controllers to different tasks is difficult to realize, as all control units have to be updated separately, also including consensus mechanisms. As a first step, this work is therefore focusing on an adaptable central control strategy for scenarios with a limited amount of involved robots. An example for such a scenario is given in Fig. 1, where the interaction of three single robots with different objectives is controlled by a central processing unit. Each robot can be abstracted to a single cohesive entity defined as an agent. An agent is defined by its objective (e.g. surveillance, landing) and its inherent constraints (e.g. dynamics, actuator limits). In cooperative control scenarios, these agents are interacting which is abstracted in this paper as coupling (e.g. collision avoidance constraints). The implemented control framework is able to adapt to such scenarios by adding and removing agents, their constraints, objectives and couplings and changing their parameters.

In order to formulate such complex control scenarios, the framework is based on *NMPC* which allows describing system constraints as equality and inequality constraints in a constrained optimization problem. A detailed review on related work about cooperative control and *NMPC* is given in section II. The advantage of being able to consider constraints comes in hand with a typically complex controller and a high computational burden. The desired on-line adaptability is further exacerbating this issues, as additional adaptation mechanisms are required. An efficient solution for such a mechanism has been presented in previous work [4], by collecting the mathematical equations of agents, constraints and couplings which are required by the solver in separate modules. The control scenario is then composed of these single exchangeable modules.

This paper is contributing an extension of the presented modularization to handle constraints with a logarithmic barrier method which is implemented as shown in section III. Furthermore, section IV is presenting the implementation structure of the framework and the utilized fast pointer based function access scheme in *C++*. As a result, the computational overhead introduced by the modularization is minimized. In addition the contributed framework implementation, uses *ROS* to trigger the framework adaptation by message events. The final contribution of the paper is the experimental validation of the event-based on-line

adaptability of the *NMPC*, shown in section V. This includes the on-line switching of dynamics, constraints and coupling constraints in a quadrotor formation flying scenario. The setup of the validation scenario is described in subsection V-A. A performance analysis and final validation of the proposed event-based *NMPC* method is given in subsection V-B. The final conclusion and future perspective of the presented work is summarized in section VI.

II. Related Work

To tackle multi agent problems, a wide variety of Distributed Model Predictive Control (*DMPC*) algorithms are available. A comprehensive summary of *DMPC* is given in [5], which covers game based, consensus based, tube and robust distributed model predictive control techniques. [6] and [7] give an overview on *NMPC* cost functions for mobile robotics regarding swarm behavior. A non-*NMPC* based framework ("CAVIS") for multi robot aerial manipulation is introduced in [8]. Also [9] and [10] are presenting open software platforms for swarm coordination. However, all these control solutions are either task specific or are lacking the desired on-line adaptability. For this reason, previous work [4] has introduced a modularization technique for *NMPC*. *NMPC* is a generic control approach to handle constrained nonlinear systems and is based on solving an optimal control problem (*OCP*) over a receding horizon. This means at each time instant the optimal controls are determined for a defined prediction horizon. The disadvantage of *NMPC* is its high computational burden, particularly regarding real-time applicability for systems with fast dynamics e.g. mobile robotics. One way to limit the computational burden of *NMPC* is to choose the control trajectory from a limited set of predefined control sequences as discussed in [7]. In this context [7] also gives various examples for *NMPC* penalty techniques to control multiple unmanned aerial vehicles (*UAV*). In contrast to predefined control sequences, [11] is proposing a gradient descent method to solve *OCP*s and offers easy to access source code. The widely spread real-time *NMPC* framework *ACADO* [12] is offering a variety of *NMPC* algorithms. Among others, the implemented sequential quadratic programming method with Gauß-Newton approximation of the Hessian offers very low computation times. The computational efficiency of *ACADO* is for example stated in a collision avoidance and aerial manipulation scenario of a quadrotor in [13]. Another promising real-time *NMPC* approach is the continuation generalized minimal residual (*CGMRES*) method [14]. Its source code is accessible via [15] and its computational efficiency has been validated in hover crafts [16], gasoline engines [17] and eco cruise control scenarios [18]. An overview on related constraint handling methods is given in [19]. A multiple-shooting derivative (*MSCGMRES*) has been developed in [20] to increase the computational stability. The resulting higher computation time due to the increased amount of optimization variables has been tackled by introducing a condensation method in [21]. The efficiency of this condensed multiple shooting continuation generalized minimal residual approach (*CMSCGMRES*) has been already proven in previous work [22], where it was evaluated in a single quadrotor collision avoidance scenario. Due to its performance in this scenario, the *CMSCGMRES* solver is applied as solver for the presented framework.

III. Modularization with Logarithmic Barrier Constraint Handling

NMPC is determining optimal controls $\mathbf{u}$ to minimize a given performance index J by predicting the system states $\mathbf{x}$ under use of a model for the system dynamics $\mathbf{f}$. The corresponding optimization problem that has to be solved in each controller update interval is an *OCP*:

$$\begin{aligned} \min_{\mathbf{u}(\cdot)} \quad & J(\mathbf{u}) = V\left(\mathbf{x}\left(t_f\right), t_f\right) + \int_{t_0}^{t_f} l\left(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau\right) \mathrm{d}\tau \qquad (1) \\ \text{u.c.} \quad & \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \tau), \qquad \mathbf{0} \geq \mathbf{c}(\mathbf{x}, \mathbf{u}, \tau) \\ & \mathbf{x}(t_0) = \mathbf{x}_0, \qquad \tau \in \left[t_0, t_f\right]. \end{aligned}$$

The performance index or cost function J typically consists of stage costs l and terminal costs V. The system behavior is predicted in a specified interval $\left[t_0, t_f\right]$ and determined at each control update interval using the current state measurement $\mathbf{x}_0$ as starting point. The model predictive control (*MPC*) loop is then closed by applying the determined optimal controls for the current time instant and shifting the problem horizon to $\tau \in \left[t_0 + \Delta t, t_f + \Delta t\right]$ under use of new state measurements.

A major advantage of *MPC* is the capability of also considering system constraints $\mathbf{c}$. On the other hand the handling of inequality constraints in *NMPC* is not trivial. In the previous publication [4], a modularization technique was introduced based on an auxiliary variable technique as described in [16]. The auxiliary variable constraint handling is transforming inequality constraints into equality constraints with the help of slack variables α [14]. Accordingly, each inequality constraint is expressed with two additional optimization variables: the slack variable and the equality constraint Lagrange multiplier. For example a robot with 4 inputs and a limitation constraint on each input yields to 12 variables to be optimized over the horizon. This leads to a high problem dimension and therewith computational burden. As alternative, the framework implementation also features logarithmic barrier constraint handling. This approach transforms each inequality constraint c to an additional stage cost term l_c with the help of a logarithmic barrier function [23]

$$0 \geq c(\mathbf{x}, \mathbf{u}, \tau) \quad \rightarrow \quad l_c = -\kappa \ln\left(-c(\mathbf{x}, \mathbf{u}, \tau)\right), \qquad (2)$$

where κ is a tuning parameter to adjust the logarithmic barrier penalty to the other cost terms. The advantage is, that no Lagrange multipliers and slack variables are needed and the dimension of the *OCP* is therefore reduced. On the other hand the *OCP* can become worse conditioned and more suboptimal, as the logarithmic barrier function tends to infinity at the point of constraint violation. However, for most mobile robotics applications this is acceptable as position constraints are typically realized as weakened constraints in the stage costs and for input limitations a

suboptimal solution is mostly affordable. For example, a control output is reaching 0.99 instead of the exact limit of 1. A further advantage is that the tuning of the logarithmic barrier function is intuitive. For these reason the logarithmic barrier method is implemented besides the auxiliary variable method in the control framework at hand.

The procedure to derive the modularization for the *OCP* with the logarithmic barrier inequality constraint handling is similar as shown in [4] for the auxiliary variable method. The considered *OCP* (1) optimality conditions are derived through the Hamiltonian. Considering each agent has its individual controls $\mathbf{u}_i$, states $\mathbf{x}_i$ and set of coupled neighbors $\mathcal{N}^i$ the Hamiltonian yields to

$$H = \sum_i^N \left[l_i(\mathbf{x}_i,\mathbf{u}_i,t) + \lambda_i^\top \mathbf{f}_i(\mathbf{x}_i,\mathbf{u}_i,t) \ldots \right. \tag{3}$$

$$- \kappa_i^\top \ln\left(-\mathbf{c}_i(\mathbf{x}_i,\mathbf{u}_i,t)\right)$$

$$- \sum_j^{|\mathcal{N}^i|} \kappa_{ij}^\top \ln\left(\mathbf{c}_{ij}\left(\mathbf{x}_i,\mathbf{x}_j,\mathbf{u}_i,\mathbf{u}_j,t\right)\right]. \tag{4}$$

For means of simplicity the handling of equality constraint has been neglected here, but can be achieved by augmenting the optimization variable vector with the equality constraint Lagrange multipliers, as shown in [4]. Each agent i is contributing to the Hamiltonian H with its individual stage costs l_i, dynamics $\mathbf{f}_i$ and constraints $\mathbf{c}_i$ additively. This holds also for the coupling constraints $\mathbf{c}_{i,j}$, which are affecting always two agents (agent i, agent j). Like the individual constraints $\mathbf{c}_i$, they are contributing to H via the logarithmic barrier method. The proposed modularization exploits this additive structure of the Hamiltonian function.

Real-time *NMPC* algorithms like *CMSCGMRES* are determining the optimal controls by solving the *OCP* first order optimality conditions. For m agents, these can be composed according to the concatenation of optimization variables $\mathbf{u} = [\mathbf{u}_1, \ldots \mathbf{u}_m]$ and state variables $\mathbf{x} = [\mathbf{x}_1, \ldots \mathbf{x}_m]$,

$$\mathbf{0} = \frac{\partial H}{\partial \mathbf{u}} = \left[\frac{\partial H}{\partial \mathbf{u}_1}^\top, \ldots \frac{\partial H}{\partial \mathbf{u}_m}^\top\right]^\top \tag{5}$$

$$\dot{\mathbf{x}} = \frac{\partial H}{\partial \lambda} = \left[\mathbf{f}_1(\mathbf{u}_1,\mathbf{x}_1,t)^\top, \ldots \mathbf{f}_m(\mathbf{u}_m,\mathbf{x}_m,t)^\top\right]^\top. \tag{6}$$

$$\dot{\lambda} = -\frac{\partial H}{\partial \mathbf{x}} = \left[-\frac{\partial H}{\partial \mathbf{x}_1}^\top, \ldots -\frac{\partial H}{\partial \mathbf{x}_m}^\top\right]^\top. \tag{7}$$

The idea is to exploit the additive structure of the Hamiltonian which is also visible in its derivatives. The modularization is exemplary executed for an element $\frac{\partial H}{\partial \mathbf{u}_i}$ of (5)

$$0 \overset{!}{=} \frac{\partial H}{\partial \mathbf{u}_i} = \frac{\partial l_i(\mathbf{x}_i,\mathbf{u}_i,t)}{\partial \mathbf{u}_i} + \lambda_i^\top \frac{\partial \mathbf{f}_i(\mathbf{x}_i,\mathbf{u}_i,t)}{\partial \mathbf{u}_i} \tag{8}$$

$$-\left(\kappa_i \backslash \mathbf{c}_i(\mathbf{x}_i,\mathbf{u}_i,t)\right)^\top \frac{\partial \mathbf{c}_i(\mathbf{x}_i,\mathbf{u}_i,t)}{\partial \mathbf{u}_i} \ldots \tag{9}$$

$$-\left(\kappa_{ij} \backslash \mathbf{c}_{ij}\left(\mathbf{x}_i,\mathbf{x}_j,\mathbf{u}_i,\mathbf{u}_j,t\right)\right)^\top \frac{\partial \mathbf{c}_{ij}(\mathbf{x}_i,\mathbf{x}_j,\mathbf{u}_i,\mathbf{u}_j,t)}{\partial \mathbf{u}_i}. \tag{10}$$

where "\" is representing an element wise division. Regarding the structure of (8)-(10), three influences can be distinguished. For example, (8) consists of the agents dynamics and stage costs which typically formulate the objective. (9) is associated with the constraints that are acting on agent i, while (10) is the influence induced by coupling constraints. In reverse conclusion, if (8) is provided for different agents, (9) for different constraints and (10) for different coupling constraints, the summands of this equation can be exchanged according to the dynamics, objective, constraints and couplings. To structure the full *OCP*, the modularization has to be also executed for the other optimality conditions (6)-(7) In case of (6) this is trivial, as it results in a concatenation of system dynamics. For (7) the modularization can be executed straight forward as for (8).

As described in [4] each of the summands (e.g. $\lambda_i^\top \frac{\partial \mathbf{f}_i(\mathbf{x}_i,\mathbf{u}_i,t)}{\partial \mathbf{u}_i}$) in (5-7) can be considered as an atomic function. These can be provided in fast compiled code and modularly added or removed from the optimality conditions. For example a change of an agent has influence on all optimality conditions, therefore it makes sense to be able to change all of its corresponding atomic functions at once. This is realized by packing them in class containers, from which the atomic functions are addressed via pointers. The fast function access via pointers is reducing the computational overhead caused by the modularization. This overlying structure of class containers of the framework will be explained in detail in the following section.

IV. Framework Structure

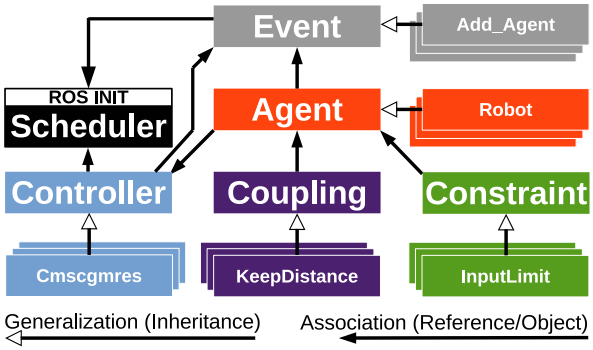


Fig. 2: Control framework structure

The principle structure of the controller package is shown in Fig. 2. It consists of five base class containers **Controller**, **Agent**, **Constraint**, **Coupling**, **Event** which provide the basic functionality for the user defined control problem. Each user agent, constraint, etc. is implemented as child class of these base class containers and therefore inheriting the interface which is required for the modular composition of the *NMPC*. The **Scheduler** class is handling the communication of the agents with the controllers.

The **Agent** class represents a generalization of user defined agents. It provides the interface for system dynamics, cost functions and their derivatives. Furthermore, each agent contains a list of pointers to constraints and/or couplings, which are acting on it. Besides the declaration of the atomic functions for the modularization, the **Agent** class provides a *ROS* communication interface. This interface allows the subscription of measurement data and publication of controls.

According to the modularization, also atomic functions of constraints and couplings can be structured in the base classes **Constraint**, respectively **Coupling**. These contain

interfaces for stage costs, final costs, equality and inequality constraints functions, as well as their derivatives. In contrast to the individual agent **Constraint**, the **Coupling** class functions are acting on two agents. This allows establishing interactions between agents and therefore the extension to arbitrary multi agent systems.

The main function of the **Controller** base class is the concatenation of optimization variables and the composition of the optimality conditions (5-7). This concatenation is based on the modularization for logarithmic barrier constraint handling of section III or the auxiliary variable method as presented in [4]. The controller contains a list of pointers to all agents that are controlled. A schematic example of the pointer access scheme to the atomic functions of the control scenario is given in Listing 1 for the composition of $\frac{\partial l}{\partial x}$. The controller iterates through all controlled agents (ptr1) and all associated constraints (ptr2) as well as couplings (ptr3).

Listing 1: Schematic pointer access scheme for the example of composing $\frac{\partial l}{\partial x}$

```
void Controller::dldx(out,t,u,x){
    //Loop over agents
    for(int i=0;i<this->agentlist.size();i++){
        //Get pointer to agent
        ptr1=agentlist_[i];
        //Get function values
        ptr1->dldx(out,t,u,x);
        //Loop over constraints in agents
        for(int j=0;j<ptr1->getConstraint_Dim();j++){
            //Get constraint pointer
            ptr2=ptr1->constraint_[j];
            //Get function values
            ptr2->dldx(out,t,u,x)
        }
        //Loop over couplings in agents
        for(int k=0;k<ptr1->getCoupling_Dim();k++){
            //Getting coupling pointer
            ptr3=ptr1->coupling_[k];
            //Get function values
            ptr3->dldx(out,t,u,x);
        }
    }
}
```

Listings 1 shows that this pointer based modularization allows a manipulation of the complete scenario by just modifying the pointer reference. The computational overhead is limited to the pointer access and iteration overhead times, as well as the allocation of auxiliary variables that store temporary results. According to this access scheme, the complete optimality conditions (5)-(7) are composed and then provided to the *CMSCGMRES* solver.

Besides the concatenation of the optimality conditions, the **Controller** class is also handling the concatenation of the optimization variables as x (states), u (controls), *udes* (target states), *xdes* (target controls), p (parameters), λ (states Lagrange multipliers), μ (equality constraint Lagrange multipliers) and under use of the auxiliary variable constraint method μ_i (inequality constraint Lagrange multipliers), α (inequality constraint slack variables).

Finally, the **event** class is the manifestation of the main advantage of the provided approach in comparison to other *NMPC* frameworks. It handles the event-triggered on-line addition and removal of the discussed containers as agents, constraints and couplings to the *OCP*. Furthermore, it allows the on-line modification of parameters. A few useful examples for this are the on-line adjustment of the state tracking penalty factors, parameters of the agent dynamics or the change of distances in formation coupling constraints. As a result the optimal control problem can be modified dynamically according to events like e.g. timers, ROS communication events, etc.

V. Experimental validation

This section is validating the performance and on-line adaptability of the event-based real-time *NMPC* framework in a real quadrotor formation flying experiment.

A. Experimental setup

The considered scenario shows a cooperative control of three *AR.Drone 2.0*© quadrotors. Each of the quadrotors is controlled by its inputs (forward-, sideward-, upward-, heading velocity)

$$\mathbf{u}_i(t) = \begin{bmatrix} u_{f,i}, & u_{s,i}, & u_{z,i}, & u_{\Psi,i} \end{bmatrix}^\top, \tag{11}$$

regarding its states ($\mathscr{W}$: global Cartesian frame, $\mathscr{V}$: vehicle frame)

$$\mathbf{x}_i = \begin{bmatrix} x_{\mathscr{W},i}, & y_{\mathscr{W},i}, & z_{\mathscr{W},i}, & \Psi_{\mathscr{W},i}, & \dot{x}_{\mathscr{V},i}, & \dot{y}_{\mathscr{V},i} \end{bmatrix}^\top, \tag{12}$$

subject to its dynamics [22]

$$\dot{\mathbf{x}}_i(t) = \mathbf{f}_i(\mathbf{x}_i, \mathbf{u}_i, t) = \begin{bmatrix} \dot{x}_{\mathscr{V},i}(t)\cos(\Psi_i) - \dot{y}_{\mathscr{V},i}\sin(\Psi_i) \\ \dot{x}_{\mathscr{V},i}(t)\sin(\Psi_i) + \dot{y}_{\mathscr{V},i}\cos(\Psi_i) \\ 1 \cdot u_{z,i}(t) \\ 0 \cdot \Psi_i(t) + 1.6 \cdot u_{\Psi,i}(t) \\ -0.5092 \cdot \dot{x}_{\mathscr{V},i}(t) + 1.458 \cdot u_{f,i}(t) \\ -0.5092 \cdot \dot{y}_{\mathscr{V},i}(t) + 1.458 \cdot u_{s,i}(t) \end{bmatrix}. \tag{13}$$

The controls of each quadrotor are limited to $\|u_i\| \leq 1$ with the help of inequality constraints [22]

$$\mathbf{0} \geq \mathbf{c}_i = \begin{bmatrix} u_{z,i}^2 - 1, & u_{\Psi,i}^2 - 1, & u_{f,i}^2 - 1, & u_{s,i}^2 - 1 \end{bmatrix}^\top. \tag{14}$$

Furthermore, between each pair of quadrotors a collision avoidance (*CA*) is implemented. This coupling forces the quadrotors to keep a minimal Euclidean d distance from each other. The coupling is realized as weakened constraint by using a sigmoid cost function to penalize the violation of the minimum distance d [22]

$$\vec{x}_i = \begin{bmatrix} x_{\mathscr{W},i}(t), y_{\mathscr{W},i}(t), z_{\mathscr{W},i}(t) \end{bmatrix}^\top \tag{15}$$

$$l_{ij}(t) = \frac{a}{1 + e^{-b\left(d^2 - \left(\vec{x}_i(t) - \vec{x}_j(t)\right)^\top \left(\vec{x}_i(t) - \vec{x}_j(t)\right)\right)}}. \tag{16}$$

Accordingly, the agents keep a minimum distance of d meters, where a is determining the cost factor and b the sigmoid slope steepness The scenario is chosen to show the switching behavior with a change of formation. For this purpose each agent is tracking the same point $\mathbf{x}^* = \begin{bmatrix} 0,0,2 \end{bmatrix}^\top$ and minimal energy consumption $\mathbf{u}^* = \mathbf{0}$ by means of stage costs

$$l_i(t, \mathbf{x}^*) = (\mathbf{x}^* - \mathbf{x}_i(t))^\top \mathbf{Q}_x (\mathbf{x}^* - \mathbf{x}_i(t)) + \mathbf{u}_i(t)^\top \mathbf{R}_u \mathbf{u}_i(t) \tag{17}$$

Starting with two quadrotor agents (ag_0, ag_1) this results in the *OCP* problem

$$\min_{\mathbf{u}} J = \int_{t_0}^{t_f} \sum_{i=0}^{1} l_i\left(\tau, [0,0,2]^\top\right) + l_{01}(\tau) \quad \mathrm{d}\tau \tag{18}$$
$$\text{s.t. } \mathbf{c}_0(\tau) \leq \mathbf{0}, \mathbf{c}_1(\tau) \leq \mathbf{0}$$
$$\mathbf{0} = \mathbf{f}_0(\mathbf{x}_0, \mathbf{u}_0, t), \mathbf{0} = \mathbf{f}_1(\mathbf{x}_1, \mathbf{u}_1, t).$$

For means of simplicity the initial states are not shown in the OCP formulation. At time t_1 a quadrotor and the corresponding constraints are added on-line to the system, triggered by a *ROS* message. The resulting OCP for three quadrotors is given by

$$\min_{\mathbf{u}} J = \int_{t_0}^{t_f} \sum_{i=0}^{2} l_i\left(\tau, [0,0,2]^\top\right) + l_{01}(\tau) + l_{12}(\tau) + l_{02}(\tau) \quad \mathrm{d}\tau \tag{19}$$
$$\text{s.t. } \mathbf{c}_0(\tau) \leq \mathbf{0}, \mathbf{c}_1(\tau) \leq \mathbf{0}, \mathbf{c}_2(\tau) \leq \mathbf{0}$$
$$\mathbf{0} = \mathbf{f}_0(\mathbf{x}_0, \mathbf{u}_0, t), \mathbf{0} = \mathbf{f}_1(\mathbf{x}_1, \mathbf{u}_1, t), \mathbf{0} = \mathbf{f}_2(\mathbf{x}_2, \mathbf{u}_2, t).$$

At time instance t_2, a second *ROS* message triggers the removal of ag_2. Accordingly, the OCP is switching again from form (19) to form (18). The resulting change of the quadrotor formation is discussed in section V-B.

To solve the given problem, a *CMSCGMRES* solver is applied under use of the logarithmic barrier constraint handling. The scenario parameters have been determined empirically to achieve a smooth system response. The state and control penalty matrices are chosen to

$$\mathbf{Q}_x = Diag\left(\left[2,2,8,3,10.5,10.5\right]\right) \tag{20}$$
$$\mathbf{R}_u = Diag\left(\left[5.5,5.5,3,3.1\right]\right). \tag{21}$$

For ease of visualization of the experiment, the penalty on the z-axis is chosen high to limit the movement mainly to the x,y-Plane. The *CA* coupling parameters are

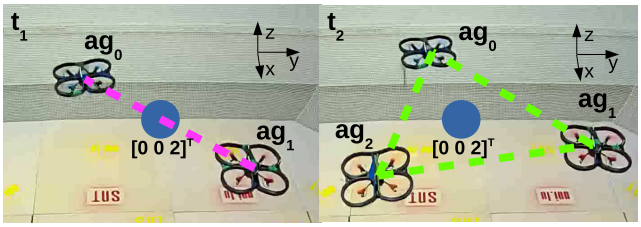
$$\mathbf{P}_{CA} = [d,a,b] = [1.5,4.0,2.0]. \tag{22}$$

Parameters of the controller are the number of discretization steps $n = 10$ of the prediction horizon $T = 3$, the desired convergence rate $\zeta = 10$, the maximal amount of iterations $k_{max} = 6$ and the minimal tolerance $r_{tol} = 0.1$ of the solver (*CMSCGMRES*). The horizon expansion factor in the initialization phase is given by $\alpha = 2$. To control the fast dynamics of the quadrotors, a short control update interval of $\Delta t = 0.1$s is chosen. The forward difference time step for the Hessian approximation is set to $\Delta t_H = 0.001$s.

B. Analysis of the Experiment

This section is analyzing the system's response to the on-line modification of the *OCP* and the performance of the proposed framework. The system behavior of the experiment is visualized in Fig. 3. At the beginning of the scenario two agents are tracking a center point $\mathbf{x}^* = [0,0,2]^\top$ (marked as blue sphere in the image centers) which would result in a collision in $\mathbf{x}^*$. The collision avoidance is inhibiting this behavior by forcing them to keep a minimum distance of $d = 1.5$m. Accordingly, both drones form a linear formation which is equidistant from the center point as shown in the image on the left of Fig. 3a. The optimal control problem for both drones is given by (18). At time $t_1 \approx 8$s the first event is provoked by a *ROS* message which adds a drone (ag_2) to the system. Accordingly, also the quadrotor input constraints ($\mathbf{c}_2$) and collision avoidance couplings (l_{02}, l_{12}) to each of the previous agents (ag_0, ag_1) are added, which results in OCP (19). As all drones are tracking the center point subject to collision avoidance constraints, this results in a triangular formation which is equidistant to the center. A second *ROS* message at t_2 is removing ag_2 and the associated constraints from the system again. Accordingly, ag_0, ag_1 are forming the original linear formation again.

(a) Video footage: t_1: 2 agents (left) vs. t_2:3 agents (right)



(b) 3D position plot with formation visualization

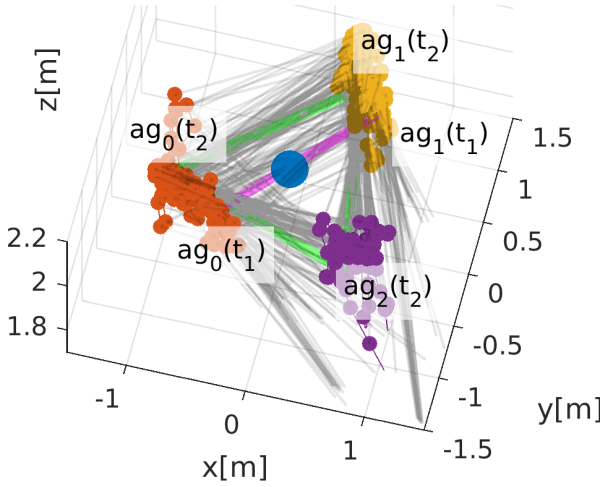


Fig. 3: Experiment: One agent with input and coupling constraints added on-line to a two agent formation

The trajectory of the agents in 3D space is given in Fig. 3b, where each agent is depicted as a small sphere. As the tracking is limited to one point and the collision avoidance is only considering the Euclidean distance of the agents, the formation can rotate freely around the tracked center point. The distance between corresponding agents at each time instance is visualized as gray lines. Before t_1, ag_2 is not within the bounds of the plotted region, as indicated by the distance connectors. In accordance to the expected behavior, Fig. 3b shows how the formation is then shifted from a linear to a triangular formation between t_1 and t_2. This validates the on-line modification of the *OCP*.

The aberration of the agent position is caused by disturbance. Particularly the mutual influence of the quadrotor airflow, including tracking and the repulsive collision avoidance coupling, is causing oscillations in the quadrotor positions.

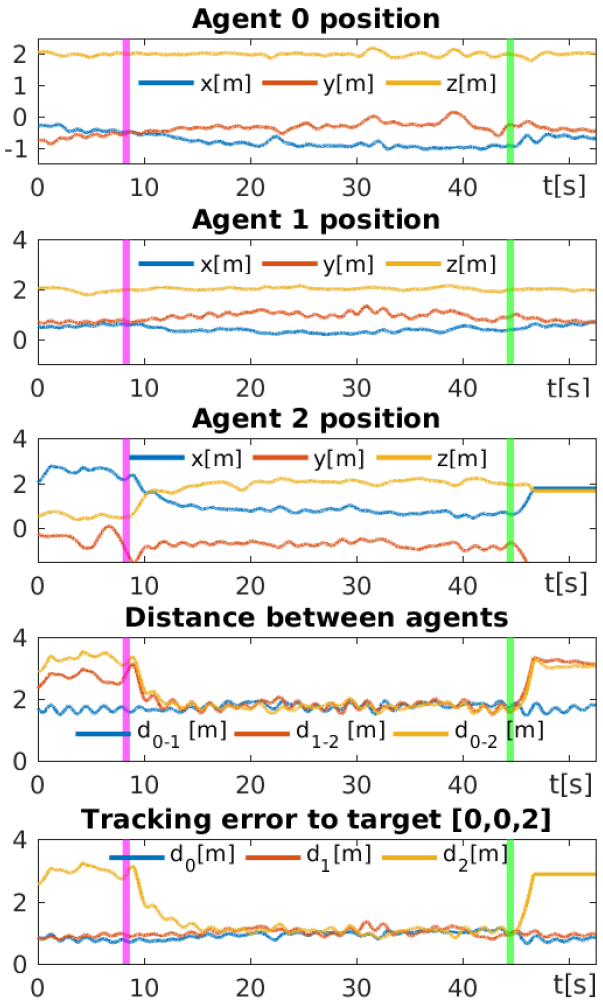


Fig. 4: Experiment: Agent position trajectories, agent distance and tracking error

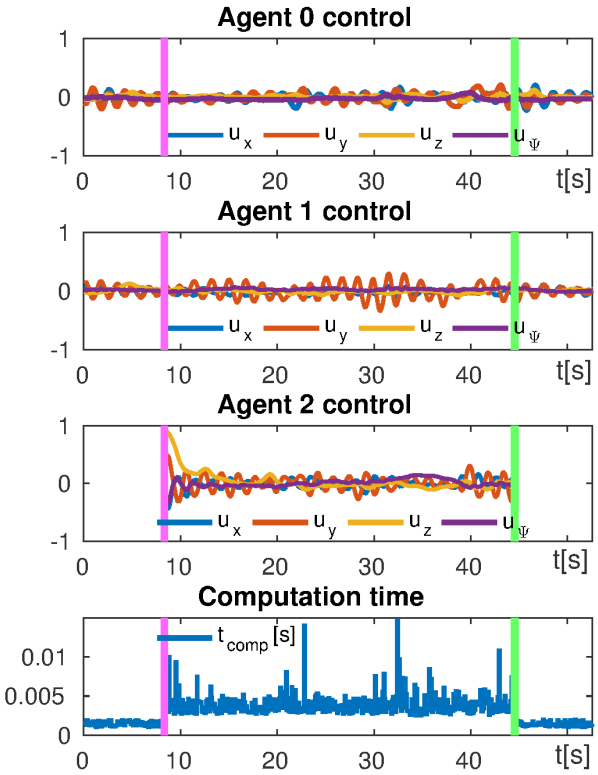


Fig. 5: Experiment: Control trajectory and computation time

These oscillations can be reduced by penalizing the *UAV* velocity states under the cost of slower trajectory changes. Another way to reduce this disturbance is to consider a disturbance model within the quadrotor model (13).

To further validate the event-based MPC switching approach, the agent positions in Fig. 4 are analyzed. The distance graph in Fig. 4 shows that at the beginning of the experiment, $\|\vec{x}_{ag_2} - \vec{x}_{ag_1}\|_2 \approx 2.5$m and $\|\vec{x}_{ag_2} - \vec{x}_{ag_0}\|_2 \approx 3.0$m. This indicates that ag_2 is not tracking the target $\mathbf{x}^*$ before t_1. At time instance t_1 (left vertical bar in the graphs), the distance of ag_2 to the other agents converges to $d = 1.5$m. The "Tracking error to target [0,0,2]" graph in Fig. 4 confirms the equidistant alignment of the agents to the target. This confirms an active *CA*-coupling and target tracking. The increase in the distance after t_2 (right vertical bar in graphs) is evidence that the position tracking is not active anymore. The agent trajectories show, that switching of tracking and couplings are working, but it does not confirm, that the input constraints are switched on-line as well. This can be extracted from the control trajectory of ag_2 in Fig. 5. As previously discussed at time instance t_1, the tracking error of ag_2 is high $\|[0,0,2]^\top - \vec{x}_{ag_2}\|_2 \approx 3$m. Accordingly, the controller response in the moment of adding the agent is high. Fig. 5 shows such behavior in the u_z control trajectory of ag_2. Due to the control limit constraints, the resulting peak does not exceed the predefined limit of $u_{z,max} = 1$. This indicates an active constraint handling. The oscillations in the controls are due to the position disturbance as discussed previously.

A major advantage of the proposed method is the low computation time as shown in Fig. 5. For the *OCP* (18) of the double agent system (ag_0,ag_1) the average computation time is $t_{c,av} = 1.5$ms, the maximum computation time is $t_{c,av} = 2.0$ms. The measurements were made on a standard notebook of type "Dell Latitue E5440 (CPU: i5-4300U with $11.29 GFLOPS/s$)". The three agent system *OCP* (19) with ag_0, ag_1 and ag_2 is solved in $t_{c,av} = 4$ms, whereas there are some single peaks up to $t_{c,max} = 14.7$ms. These single peaks are expected to be outliers that are caused by delay due to other CPU processes, but will be investigated further. At first sight, adding one agent causing a doubling of the computation time might look over proportional, however the amount of *CA* couplings is increased by 2.

The average computation time of $t_{c,av} = 4$ms for the three drone scenario with a control update interval of $\Delta t = 0.1$s yields to an average computational load of 4% on a standard notebook *CPU*. This states the computational efficiency of the proposed event-based real-time *NMPC* approach, under use of the proposed constraint handling technique, modularization, framework structure and *CMSCGMRES* solver.

VI. CONCLUSIONS

This paper presents the implementation of a novel central event-based real-time *NMPC* framework which offers the flexibility to modify the control scenario on-line, while maintaining low computation times. The framework targets real-time control of dynamically changing control scenarios, which makes it particularly interesting for single and multi robot applications, as well as fault-tolerant control.

To achieve the desired flexibility, the paper is presenting the implementation structure to a previously presented modularization of the *OCP* into atomic functions. To be able to modify complete *OCP* modules, like agents, constraints and couplings, the corresponding atomic functions have been structured into base class containers. Furthermore, the modularization is extended by a logarithmic barrier inequality constraint handling, to offer an alternative to the previously shown auxiliary variable method which results in high dimensional problems. The availability of the atomic functions in compiled $C/C++$ code, together with an access scheme via pointers, results in low computation times.

The performance and flexibility of the proposed event-based real-time *NMPC* framework has been validated in a quadrotor formation flying scenario. The experiment shows a linear flight formation of two *AR.Drone 2.0*© quadrotors that is extended to a three quadrotor triangular formation, triggered by a *ROS* message event. The drone trajectories have confirmed the on-line switching of quadrotor dynamics and objectives (tracking), collision avoidance couplings and input limitation constraints. The average computation time of $t_{c,av} = 1.5$ms for the two drone scenario and $t_{c,av} = 4$ms for the three drone scenario states the computational efficiency of the proposed event-based *NMPC* framework. The framework source code is available under [2].

Future work will focus on the implementation of distributed *NMPC* methods to make also use of event-based *NMPC* on large-scale systems. Further implementations will also address additional inequality constraint handling techniques. To evaluate the performance of these techniques an extensive benchmark and their mathematical evaluation is a future field of interest. Furthermore, the effects of the event-based *NMPC* adaptation have to be analyzed regarding problem feasibility and control performance.

REFERENCES

[1] M. Quigley, K. Conley, B. P. 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*, 2009.

[2] J. Dentler, "An event-based real-time nonlinear model predictive control framework," http://wiki.ros.org/denmpc, http://github.com/snt-robotics/denmpc, 2016, accessed: 2017-06-21.

[3] M. F. E. Rohmer, S. P. N. Singh, "V-rep: a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[4] J. Dentler, S. Kannan, M. A. Olivares-Mendez, and H. Voos, "A modularization approach for nonlinear model predictive control of distributed fast systems," in *2016 24th Mediterranean Conference on Control and Automation (MED)*, 2016, pp. 292–297.

[5] R. R. Negenborn and J. M. Maestre, *On 35 Approaches for Distributed MPC Made Easy*. Dordrecht: Springer Netherlands, 2014, pp. 1–37.

[6] T. Gorecki, H. Piet Lahanier, J. Marzat, and M. Balesdent, "Cooperative guidance of UAVs for area exploration with final target allocation."

[7] S. Bertrand, J. Marzat, H. Piet-Lahanier, A. Kahn, and Y. Rochefort, "MPC Strategies for Cooperative Guidance of Autonomous Vehicles," *AerospaceLab*, no. 8, pp. 1–18, 2014. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01103195

[8] "Cavis: a control software architecture for cooperative multi-unmanned aerial vehicle-manipulator systems," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 1108 – 1113, 2014, 19th IFAC World Congress.

[9] M. Furci, G. Casadei, R. Naldi, R. G. Sanfelice, and L. Marconi, "An open-source architecture for control and coordination of a swarm of micro-quadrotors," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, 2015, pp. 139–146.

[10] X. Dong, B. M. Chen, G. Cai, H. Lin, and T. H. Lee, "Development of a comprehensive software system for implementing cooperative control of multiple unmanned aerial vehicles," in *2009 IEEE International Conference on Control and Automation*, 2009, pp. 1629–1634.

[11] K. Graichen and B. Käpernick, "A real-time gradient method for nonlinear model predictive control," 2012.

[12] D. Ariens, M. Diehl, H. Joachim, B. Houska, Ferreau, F. Logist, R. Quirynen, and M. Vukov, "ACADO Toolkit User's Manual," 2015, accessed: 2015-09-11.

[13] M. Geisert and N. Mansard, "Trajectory generation for quadrotor based systems using numerical optimal control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 2958–2964.

[14] "A continuation/gmres method for fast computation of nonlinear receding horizon control," *Automatica*, vol. 40, no. 4, pp. 563 – 574, 2004.

[15] T. Ohtsuka, "symlab: Cgmres," http://www.symlab.sys.i.kyoto-u.ac.jp/ ohtsuka/code/index.htm, accessed: 2015-09-4.

[16] H. Seguchi and T. Ohtsuka, "Nonlinear receding horizon control of an underactuated hovercraft," vol. 13, no. 3-4. Wiley Online Library, 2003, pp. 381–398.

[17] "Continuation/gmres method based nonlinear model predictive control for ic engines," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 5697 – 5702, 2014, 19th IFAC World Congress.

[18] S. A. Sajadi-Alamdari, H. Voos, and M. Darouach, "Nonlinear model predictive extended eco-cruise control for battery electric vehicles," in *2016 24th Mediterranean Conference on Control and Automation (MED)*, 2016, pp. 467–472.

[19] M. Huang, H. Nakada, Butts, K. R., and I. V. Kolmanovsky, "Nonlinear Model Predictive Control of a Diesel Engine Air Path: A Comparison of Constraint Handling and Computational Strategies."

[20] Y. Shimizu, T. Ohtsuka, and M. Diehl, "Nonlinear receding horizon control of an underactuated hovercraft with a multiple-shooting-based algorithm," in *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, 2006, pp. 603–607.

[21] Y. Shimizu, T. Ohtsuka, and M. Diehl, "A real-time algorithm for nonlinear receding horizon control using multiple shooting and continuation/krylov method," *International Journal of Robust and Nonlinear Control*, vol. 19, no. 8.

[22] J. Dentler, S. Kannan, M. A. Olivares-Mendez, and H. Voos, "A real-time model predictive position control with collision avoidance for commercial low-cost quadrotors," in *2016 IEEE Multi-Conference on Systems and Control (MSC), Buenos Aires*, September 2016.

[23] R. A. Polyak, "Barrier functions and their modifications," in *Encyclopedia of Operations Research and Management Science*, S. I. Gass and M. C. Fu, Eds. Boston, MA: Springer US, 2013, pp. 102–106.