Distributed MPC for multi-vehicle systems moving in formation[☆]

Ruben Van Parys[a,*], Goele Pipeleers[a]

[a]*MECO Research Group, Department of Mechanical Engineering, Division PMA, KU Leuven, BE-3001 Leuven, Belgium.*

Abstract

This work presents a novel distributed model predictive control (DMPC) strategy for controlling multi-vehicle systems moving in formation. The vehicles' motion trajectories are parameterized as polynomial splines and by exploiting the properties of the B-spline basis functions, constraints on the trajectories are efficiently enforced. The computations for solving the resulting optimization problem are distributed among the agents by the Alternating Direction Method of Multipliers (ADMM). In order to reduce the computation time and the amount of inter-vehicle interaction, only one ADMM iteration is performed per control update. In this way the method converges over the subsequent control updates. Simulations for various nonholonomic vehicle types and an experimental validation on in-house developed robotic platforms prove the capability of the proposed approach. A supporting software toolbox is provided that implements the proposed approach and that facilitates its use.

Keywords: Distributed Model Predictive Control (DMPC), Nonholonomic multi-vehicle system, Alternating Direction Method of Multipliers (ADMM), B-spline.

1. Introduction

Boosted by enhancements in communication technologies and computational power, networked multi-vehicle systems have received increasing attention over the last decades. A particular application hereof is formation control of multi-vehicle systems, which forms the basis for applications such as cooperative transportation by small automated guided vehicles or cooperative surveillance. Furthermore, it is well known that the flight efficiency substantially increases when aerial vehicles fly in close formation [1].

State-of-the-art formation control approaches for unmanned vehicles are divided in three main groups: leader-follower techniques [2, 3], virtual structure approaches [4, 5] and behavioral methods [6, 7]. Recent research in these areas mainly focuses on formation stabilization [8, 9] and formation following a predefined path [10, 11]. Integrating motion planning in the formation control structure is typically achieved by separating the problem in (i) finding a trajectory for the (virtual) leader and (ii) controlling the other vehicles to attain a desired relative position with respect to the leader [12]. This architecture is however not robust against failures of the leader. Therefore, this paper aims for an approach in which all members of the formation are equal. This implies that each vehicle searches for its own trajectory. By allowing communication, the vehicles are able to adapt their trajectories in order to satisfy the formation constraints. Such distributed control structure benefits from the flexibility to add or discard agents from the controlled multi-agent system, the possibility to hide local information and the ability to choose and optimize the information flow between the agents. An appealing framework allowing for such architecture is distributed model predictive control (DMPC). It can explicitly address input and state constraints, account for multiple control objectives, and incorporate forecasts of disturbances. Moreover, DMPC distributes the computational load of solving the control problem among the different agents. DMPC has been a very active research area since the end of the 1990s. The reader is referred to [13, 14, 15, 16] for an overview and comparison of various existing approaches.

The control of multi-vehicle systems is in general a complex problem. Various existing MPC approaches have only been applied to the use of linear vehicle dynamics or lack the flexibility to add arbitrary (nonconvex) constraints to the problem formulation such as collision avoidance constraints [17, 18]. Approaches considering realistic nonconvex multi-vehicle problems are mostly limited to solving offline optimal control problems [19, 20]. Decoupling the multi-vehicle control problem efficiently imposes an extra difficulty. Existing DMPC strategies typically solve in every update cycle an optimization problem in a distributed

[☆]This work benefits from KU Leuven-BOF PFV/10/002 Centre of Excellence: Optimization in Engineering (OPTEC), from the Belgian Programme on Interuniversity Attraction Poles, initiated by the Belgian Federal Science Policy Office (DYSCO), from the project G0C4515N of the Research Foundation-Flanders (FWO-Flanders) and the KU Leuven Research project C14/15/067: B-spline based certificates of positivity with applications in engineering. Ruben Van Parys is a PhD fellow of FWO-Flanders. The MECO Research Group is an associated research lab of Flanders Make.

[*]Corresponding author

Email address: `ruben.vanparys@kuleuven.be` (Ruben Van Parys)

fashion [21, 22]. This generally involves multiple iterations, each of which requires solving local optimization problems and substantial communication between neighboring agents. This often results in too slow update rates in practice as the control law should be implemented on the vehicle's embedded hardware, which has restricted computational power and communication capabilities.

This paper aims at reducing the existing gap in the literature by presenting a novel DMPC strategy for controlling multi-vehicle systems. The approach focuses on realistic complex problems, including nonlinear vehicle dynamics and collision avoidance constraints. It applies to a particular class of vehicles, including holonomic vehicles, quadrotors and differential wheeled robots. Although various types of vehicle interaction can be incorporated, this paper focuses on vehicles moving in formation.

The approach is based on two main ingredients that allow to solve the resulting optimal control problem in an efficient manner. First, the multi-vehicle problem is decoupled such that the computational load can be distributed over the agents. This is achieved by applying the Alternating Direction Method of Multipliers (ADMM) [23]. In order to reduce the amount of communication, an updating scheme is proposed that solves only one ADMM iteration per control update. Second, a spline parameterization for the vehicles' motion trajectories and a related enforcement of constraints on these trajectories allow an efficient reformulation of an agent's local subproblem [24, 25, 26]. This further reduces the computational load of one control update. Although no formal stability proof is provided, various numerical results and an experimental validation demonstrate that the ADMM iterations converge over the subsequent control updates. These updates are performed at a sufficiently fast rate with only a limited loss of optimality. As a complement to the paper, a software toolbox is provided that implements the proposed approach and that forms a user-friendly interface for modeling and simulating the considered problems [27]. Furthermore, additional illustrative examples are supplied in the toolbox.

Section 2 describes the considered vehicle types and the multi-vehicle control problem. Section 3 shows how this problem is reformulated in a small-scale optimization problem, how it is solved in receding-horizon and how it is decoupled over the agents. The proposed approach is further analyzed and illustrated with simulation and experimental examples in Section 4. Finally, Section 5 draws concluding remarks. A preliminary version of this paper considering only linear vehicle dynamics in simulation was presented in [28].

2. Problem formulation

This section describes the class of vehicles examined in this work. Afterwards the multi-vehicle optimal control problem is presented.

2.1. Vehicle description

The vehicles considered in this work are described by their states $q(t) \in \mathbb{R}^{n_q}$, inputs $u(t) \in \mathbb{R}^{n_u}$, ordinary differential equation

$$\dot{q} = f(q, u)\,, \tag{1}$$

and state and input constraints over the considered time horizon

$$e(q(t), u(t)) \geq 0\,, \quad \forall t \in [0, T]\,. \tag{2}$$

In order to describe the vehicle dynamics in a numerical optimization framework, a classical collocation approach would parameterize the input u as a piecewise polynomial. The state q is then approximately determined from u by using a numerical integration scheme. Constraints on input and states are then imposed on discrete points in time [29]. In contrast, this work uses a formulation that implies an exact representation of the system dynamics (1) and that guarantees satisfaction of constraints (2) over the whole time horizon. Therefore the system is described by means of output trajectories $y(t) \in \mathbb{R}^{n_u}$ similar as in [30, 31]. These output trajectories are assumed to satisfy two requirements. First, it is assumed that the system is differentially flat [32] sucht that the input u and state q can be determined from the output y, its derivatives and anti-derivatives:

$$\begin{aligned} u &= \psi_u(y, y^{(1)}, \ldots, y^{(r)})\,, \\ q &= \psi_q(y^{(-p)}, \ldots, y, \ldots, y^{(r)})\,, \end{aligned} \tag{3}$$

where r and p are positive integers. Second, substituting (3) in the state and input constraints (2), is assumed to result in constraints that are polynomial in y, its derivatives and anti-derivatives:

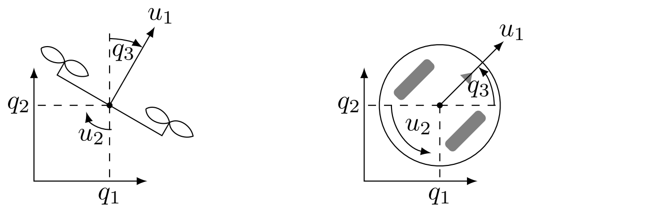
$$e(q, u) \geq 0 \iff h(y^{(-p)}, \ldots, y^{(r)}) \geq 0\,,$$

where $h(\cdot)$ is polynomial. This assumption allows to enforce constraints at all time instances of the considered time horizon when using a spline parameterization for y. This is further explained in Section 3.1.

The above-mentioned assumptions are readily verified for linear holonomic vehicles, but also hold for various nonlinear vehicle models. An example hereof is illustrated in Figure 1a, which represents a first principles quadrotor model in the vertical plane. The quadrotor is controlled by two inputs, the total thrust acceleration u_1 and the pitch rate u_2, and has three degrees of freedom, the horizontal and vertical position q_1 and q_2 and the pitch angle q_3. The equations of motion are

$$\begin{aligned} \ddot{q}_1 &= u_1 \sin q_3\,, \\ \ddot{q}_2 &= u_1 \cos q_3 - g\,, \\ \dot{q}_3 &= u_2\,, \end{aligned} \tag{4}$$

where g denotes the gravitational acceleration. This system is differentially flat with flat output $(y_1, y_2) = (q_1, q_2)$. Indeed, from (4) it follows that the third state q_3 and both



(a) 2D quadrotor (b) Differential wheeled robot

Figure 1: Examples of nonlinear vehicle models that allow polynomial constraints.

inputs can be written as a function of these outputs and their derivatives:

$$\begin{aligned} q_3 &= \arctan \frac{\ddot{y}_1}{\ddot{y}_2 + g}\,, \\ u_1 &= \sqrt{\ddot{y}_1^2 + (\ddot{y}_2 + g)^2}\,, \\ u_2 &= \frac{\dddot{y}_1(\ddot{y}_2 + g) - \ddot{y}_1 \dddot{y}_2}{\ddot{y}_1^2 + (\ddot{y}_2 + g)^2}\,. \end{aligned}$$

Bounds on the inputs can be formulated polynomially by squaring the constraint on u_1 and multiplying the constraint on u_2 by its (nonnegative) denominator. When replacing constraints on q_3 by constraints on $\tan q_3$ and multiplying them by $(\ddot{y}_2 + g)$, these become polynomial as well. Note that in the above derivations it is assumed that $u_1 \geq 0$ and $\ddot{y}_2 + g > 0$.

Another example of a nonlinear vehicle model that allows polynomial constraints is a differential wheeled robot, illustrated in Figure 1b. Its inputs are the tangential velocity u_1 and rotational velocity u_2. Its states are the position q_1, q_2 and orientation q_3, and the model is described by

$$\begin{aligned} \dot{q}_1 &= u_1 \cos q_3\,, \\ \dot{q}_2 &= u_1 \sin q_3\,, \\ \dot{q}_3 &= u_2\,. \end{aligned} \tag{5}$$

When using a similar approach as with the two-dimensional quadrotor, the transformed system would suffer from singularities when $u_1 = 0$. Furthermore, the relative formation constraints, introduced in Section 4.2, would not allow a polynomial formulation. Therefore another approach is used. The model is rewritten by introducing $z = \tan \frac{q_3}{2}$ and using relations

$$\cos q_3 = \frac{1 - z^2}{1 + z^2}\,, \quad \sin q_3 = \frac{2z}{1 + z^2}\,. \tag{6}$$

After substituting $\tilde{u}_1 = \frac{u_1}{1+z^2}$, the first two lines of the model (5) are transformed to

$$\dot{q}_1 = \tilde{u}_1(1 - z^2)\,, \quad \dot{q}_2 = 2\tilde{u}_1 z\,.$$

The states and inputs can be expressed as a function of $(y_1, y_2) = (\tilde{u}_1, z)$ as follows:

$$\begin{aligned} q_1(t) &= \int_0^t y_1(\tau)(1 - y_2(\tau)^2)d\tau + q_1(0)\,, \\ q_2(t) &= \int_0^t 2y_1(\tau)y_2(\tau)d\tau + q_2(0)\,, \\ q_3 &= 2\arctan y_2\,, \\ u_1 &= y_1(1 + y_2^2)\,, \\ u_2 &= \frac{2\dot{y}_2}{1 + y_2^2}\,. \end{aligned}$$

When replacing bounds on q_3 by constraints on $y_2 = \tan \frac{q_3}{2}$, all state and input constraints can be written as polynomials in y, its derivatives and anti-derivatives. Note that this approach does not require u_1 to be positive and allows backwards driving.

2.2. Multi-vehicle optimal control problem

The problems considered in this work search for state and input trajectories, both included in $y_i(\cdot)$, for N different vehicles i in order to steer them from an initial condition, at $t = 0$, to a terminal condition, at $t = T$. Optimal trajectories are obtained by minimizing the sum of all vehicles' objectives J_i while respecting their input and state constraints h_i, which include collision avoidance constraints. In addition the vehicles should move in formation. This is enforced by constraints $g_{i,j}$ which imply relations between the outputs y_i and y_j of respectively a vehicle i and its neighbor j. As illustrated in Section 4, this formation can be described in absolute or relative terms, depending on the definition frame. With $\mathcal{N}_i$ the neighbor set of an agent i, this multi-vehicle problem generally translates into an optimization problem of the following form:

$$\begin{aligned} \underset{y_i(\cdot),\, i=1,\dots,N}{\text{minimize}} \quad & \sum_{i=1}^{N} J_i(y_i) \\ \text{subject to} \quad & y_i^{(j)}(0) = y_{0,i}^{(j)}\,, \quad j \in \{0, \dots, r-1\} \\ & y_i^{(j)}(T) = y_{T,i}^{(j)}\,, \quad j \in \{0, \dots, r-1\} \\ & h_i(y_i^{(-p)}(t), \dots, y_i^{(r)}(t)) \geq 0 \\ & g_{i,j}(y_i(t), y_j(t)) = 0, \quad \forall j \in \mathcal{N}_i \\ & \forall t \in [0, T], \quad \forall i \in \{1, \dots, N\}\,. \end{aligned} \tag{7}$$

Collision avoidance constraints, included in h_i, are function of a vehicle's position and orientation. Position and orientation are states of a vehicle and are assumed to depend polynomially on the output trajectories y_i, their derivatives and anti-derivatives. The collision avoidance constraints are formulated by a technique described in [25, 26], which is recapitulated in Appendix A.

3. Spline-based DMPC

This section describes the proposed approach for solving the multi-vehicle formation control problem in an efficient and distributed fashion. First, problem (7) is translated into a nonlinear program by adopting a spline parameterization for the motion trajectories and an efficient enforcement of constraints on these trajectories. Second, a scheme is proposed for solving the resulting problem in receding-horizon. Finally, this scheme is further adapted to a DMPC strategy in order to distribute the computational load among the vehicles.

3.1. Spline parameterization

Problem (7) is infinite dimensional, comprising both infinitely many optimization variables and constraints, as the optimization variables $y_i(\cdot)$ are functions and constraints on them are enforced at all time instances. This problem is transformed into a small-scale optimization problem along the lines of [24]. To cope with the infinitely many optimization variables, the trajectories $y_i(\cdot)$ are approximated as piecewise polynomials and are parameterized in a B-spline basis [33]:

$$\hat{y}_i(t) = \sum_{l=1}^{n} \mathsf{y}_{i,l} b_l(t) = \mathsf{y}_i^T b(t)\,,$$

with B-spline basis $b = [b_1, \dots, b_n]^T$ and B-spline coefficients $\mathsf{y}_i = [\mathsf{y}_{i,1}, \cdots, \mathsf{y}_{i,n}]^T$. The main reason for adopting the B-spline basis is the so-called convex hull property: as the B-splines are positive and sum up to 1, a spline is always contained in the convex hull of its B-spline coefficients. This way, bounds on a spline function can be enforced by imposing them on the coefficients:

$$\mathsf{y}_i \geq 0 \Rightarrow \hat{y}_i(t) \geq 0\,, \forall t \in [0, T]\,.$$

Because any polynomial function of a spline is itself a spline, also polynomial constraints on spline trajectories can be relaxed in the same way. When $h_i(\cdot)$ in (7) is a polynomial, one can write

$$\begin{aligned} &h_i(\hat{y}_i^{(-p)}(t), \dots, \hat{y}_i^{(r)}(t)) \\ &\quad = h_i(\sum_{l=1}^{n} \mathsf{y}_{i,l} b_l^{(-p)}(t), \dots, \sum_{l=1}^{n} \mathsf{y}_{i,l} b_l^{(r)}(t)) \\ &\quad = \sum_{l=1}^{m} \mathsf{h}_{i,l}(\mathsf{y}_i) \tilde{b}_l(t) \\ &\quad = \mathsf{h}_i(\mathsf{y}_i)^T \tilde{b}(t)\,, \end{aligned}$$

with $\tilde{b} = [\tilde{b}_1, \dots, \tilde{b}_m]^T$ and $\mathsf{h}_i(\mathsf{y}_i) = [\mathsf{h}_{i,1}(\mathsf{y}_i), \dots, \mathsf{h}_{i,m}(\mathsf{y}_i)]^T$ the basis and coefficients of the resulting spline. The convex hull property implies that

$$\mathsf{h}_i(\mathsf{y}_i) \geq 0 \Rightarrow h_i(\hat{y}_i^{(-p)}(t), \dots, \hat{y}_i^{(r)}(t)) \geq 0\,, \forall t \in [0, T]\,,$$

which allows to replace the right-hand side by the more strict left-hand side. Note that this replacement is however conservative. Systematic approaches to reduce the conservatism are discussed in [24].

Algorithm 1 Spline-based MPC

```
1: Repeat every $\Delta T$: $k = 0, 1, \dots$
2:     Extract $q_i^k(t)$, $u_i^k(t)$ from $\mathsf{y}_i^k$
3:     Vehicle $i$ starts following trajectories $q_i^k(t)$, $u_i^k(t)$
4:     Estimate $\hat{q}_i^k = q_i^k((k+1)\Delta T)$, $\hat{u}_i^k = u_i^k((k+1)\Delta T)$
5:     Update horizon and compute $\tilde{\mathsf{y}}_i^k$
6:     Compute $\mathsf{y}_i^{k+1}$ by solving (8), using $\hat{q}_i^k$, $\hat{u}_i^k$ as initial
       conditions and $\tilde{\mathsf{y}}_i^k$ as hot-start
7: Until target reached
```

Following these steps, problem (7) is reformulated in terms of the spline coefficients y_i of the agents' trajectories:

$$\begin{aligned} \underset{\mathsf{y}_i,\, i=1,\dots,N}{\text{minimize}} \quad & \sum_{i=1}^{N} \mathsf{J}_i(\mathsf{y}_i) \\ \text{subject to} \quad & \mathsf{y}_i \in \mathsf{Y}_i \\ & \mathsf{g}_{i,j}(\mathsf{y}_i, \mathsf{y}_j) = 0\,, \quad \forall j \in \mathcal{N}_i \\ & \forall i \in \{1, \dots, N\}\,. \end{aligned} \tag{8}$$

The objective J_i and constraint functions $\mathsf{g}_{i,j}$ are formulated in terms of spline coefficients of $\hat{y}_i$. The set Y_i represents the initial and terminal condition constraints and state and input constraints h_i for an agent i.

3.2. Spline-based MPC

This subsection describes how the multi-vehicle problem (8) is solved in an MPC fashion. Hence, at every sample period, an optimal control problem needs to be solved over an updated control horizon, starting from the estimates of the current vehicles' states and inputs. As in our optimal control approach the solution of the optimization problem is formulated as a spline, care must be taken in transferring the solution of the previous MPC iteration to a hot-start for the subsequent iteration.

The overall MPC procedure is summarized in Algorithm 1. At the start of an MPC iteration k, a vehicle starts following its state and input trajectories computed during the previous iteration (steps 2-3). Meanwhile, the initial conditions for the current optimal control problem are determined (step 4), the trajectories of the previous iteration are transformed into a hot-start for the current problem (step 5), and the optimal control problem (8) is solved (step 6).

To account for the computational delay of an MPC iteration, the approach of [34] is adopted to start the optimal control problem from a state and input estimate $\hat{q}_i^k$, $\hat{u}_i^k$ at the time the control signal will be applied. As a trajectory computed during the k^{th} iteration is applied at the start of iteration $k+1$, $\hat{q}_i^k$ and $\hat{u}_i^k$ are estimates for respectively $q_i^k((k+1)\Delta T)$ and $u_i^k((k+1)\Delta T))$ (step 4). These estimates are used to determine the initial conditions for y_i in the problem of step 6.

An updated control horizon and a corresponding spline basis are defined during step 5. In order to use the results

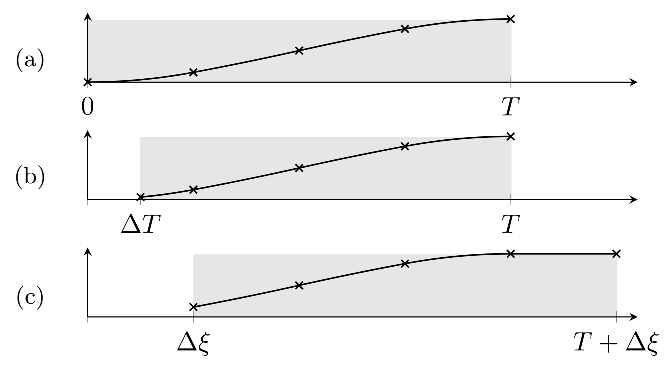


Figure 2: First three updates of the control horizon, represented by the gray area. The nominal horizon time and control period are represented by respectively T and ΔT while $\Delta\xi = 2\Delta T$ is the nominal width of the polynomial interval, i.e. the time interval between two consecutive breakpoints. The crosses indicate the breakpoints. When ΔT is smaller than the first polynomial interval, the latter is shrunk over ΔT, as in (b). When this is not the case, the first polynomial interval is discarded and an extra interval of length $\Delta\xi$ is added at the end, over which the previous computed trajectory is extrapolated, as in (c).

computed in a previous iteration as a hot-start for the current iteration, these results, which correspond to spline trajectories, are first expressed in the new basis. This means that coefficients $\tilde{\mathsf{y}}_i^k$ are computed such that

$$\sum_{l=1}^{n} \mathsf{y}_{i,l}^k b_l^{k-1}(t) = \sum_{l=1}^{n} \tilde{\mathsf{y}}_{i,l}^k b_l^k(t), \quad \forall t \in \mathcal{T}^k ,$$

where $\mathcal{T}^k$ is the control horizon considered at iteration k and $b^k = [b_1^k, \ldots, b_n^k]^T$ the corresponding B-spline basis. These transformed coefficients are then used as hot-start for the problem solved in step 6.

The way of updating the control horizon cannot be chosen freely. On the one hand it should be possible to express the future part of previous trajectories in the new basis. As a spline consists of polynomial pieces glued together with some continuity conditions at certain breakpoints, it is required to preserve the location of future breakpoints of b^{k-1} in b^k. On the other hand it is desired to keep the dimension of the bases constant over the updates. Practically this allows building the optimization problem in advance, which is a time-consuming task. Both desires are fulfilled by using the update scheme illustrated in Figure 2.

3.3. ADMM based Distributed MPC

Step 6 of Algorithm 1 requires the solution of the coupled problem (8). In this problem, many of the constraints only apply to one specific agent while the objective is composed of the individual objectives of the different agents. The only coupling is imposed by the interaction constraint $\mathsf{g}_{i,j}$. This subsection describes how the problem can be decoupled in order to distribute the computational load of solving the optimal control problem among the agents. The decoupling is derived using techniques from distributed optimization. Various approaches were compared with respect to convergence speed. It turns out that the Alternating Direction Method of Multipliers (ADMM) [23] generally results in the best performance. The underlying derivations are formulated for ADMM but can equivalently be listed for other approaches.

Before applying ADMM on problem (8), the problem is first reformulated as the following equivalent one:

$$\begin{aligned} \underset{\mathsf{y}_i,\,\mathsf{z}_i,\,\mathsf{z}_{i,j},\,i=1,\ldots,N}{\text{minimize}} \quad & \sum_{i=1}^{N} \mathsf{J}_i(\mathsf{y}_i) \\ \text{subject to} \quad & \mathsf{y}_i \in \mathsf{Y}_i \\ & \mathsf{g}_{i,j}(\mathsf{z}_i, \mathsf{z}_{i,j}) = 0\,, \quad \forall j \in \mathcal{N}_i \\ & \mathsf{y}_i = \mathsf{z}_i\,, \quad \mathsf{y}_j = \mathsf{z}_{i,j}\,, \quad \forall j \in \mathcal{N}_i \\ & \forall i \in \{1, \ldots, N\}\,, \end{aligned} \tag{9}$$

i.e. for each vehicle i slack variables z_i and $\mathsf{z}_{i,j}$ are introduced that match respectively a vehicle's own trajectory y_i and the trajectory of its neighbor y_j. Furthermore, each formation constraint is replicated for the two corresponding agents, i.e. when a vehicle i defines its position relative to vehicle j by means of constraint $\mathsf{g}_{i,j}$, vehicle j define its position with respect to i with a compatible constraint $\mathsf{g}_{j,i}$. This implies bi-directional interactions:

$$j \in \mathcal{N}_i \Leftrightarrow i \in \mathcal{N}_j. \tag{10}$$

The idea behind ADMM is to solve an appropriate dual problem of (9). There are many ways to formulate this problem, depending on which constraints are dualized. Based on rules formulated in [19], which aim at reducing the duality gap, only the constraints $\mathsf{y}_i = \mathsf{z}_i$ and $\mathsf{y}_j = \mathsf{z}_{i,j}$ are dualized. ADMM makes use of the augmented Lagrangian function which adds an extra quadratic penalty term in order to improve robustness and to yield milder convergence assumptions. For problem (9) this function is

$$\begin{aligned} \mathcal{L}_\rho &= \sum_{i=1}^{N} \Big(\mathsf{J}_i(\mathsf{y}_i) + \lambda_i^T(\mathsf{y}_i - \mathsf{z}_i) + \frac{\rho}{2}\|\mathsf{y}_i - \mathsf{z}_i\|_2^2 \\ &\qquad + \sum_{j\in\mathcal{N}_i} \big(\lambda_{i,j}^T(\mathsf{y}_j - \mathsf{z}_{i,j}) + \frac{\rho}{2}\|\mathsf{y}_j - \mathsf{z}_{i,j}\|_2^2\big)\Big)\,, \\ &= \sum_{i=1}^{N} \mathcal{L}_{\rho,i}(\mathsf{y}_i, \mathsf{z}_i, \lambda_i, \mathsf{y}_j, \mathsf{z}_{i,j}, \lambda_{i,j})\,, \\ &= \sum_{i=1}^{N} \mathcal{L}_{\rho,i}(\mathsf{y}_i, \mathsf{z}_i, \lambda_i, \mathsf{y}_i, \mathsf{z}_{j,i}, \lambda_{j,i})\,, \end{aligned}$$

where λ_i and $\lambda_{i,j}$ represent the dual variables associated with the dualized constraints. The last equality is true because of the bi-directional interaction (10). The dual function is written as:

$$q(\lambda_i, \lambda_{i,j}) = \inf_{\substack{\forall i:\, \mathsf{y}_i \in \mathsf{Y}_i \\ \forall i, \forall j\in\mathcal{N}_i:\, \mathsf{g}_{i,j}(\mathsf{z}_i,\mathsf{z}_{i,j})=0}} \mathcal{L}_\rho\,.$$

By maximizing this dual function, the optimum is found. ADMM solves this dual problem by using gradient ascent. The gradient of q with respect to the dual variables is evaluated by first searching for y_i^+, z_i^+ and $\mathsf{z}_{i,j}^+$ that minimize $\mathcal{L}_\rho$. In this way, the gradient is evaluated as:

$$\nabla_{\lambda_i} q = \mathsf{y}_i^+ - \mathsf{z}_i^+ \,,$$
$$\nabla_{\lambda_{i,j}} q = \mathsf{y}_j^+ - \mathsf{z}_{i,j}^+ \,.$$

In ADMM, the search for y_i^+, z_i^+ and $\mathsf{z}_{i,j}^+$ is split in two consecutive steps. First $\mathcal{L}_\rho$ is minimized over y_i and in a second step it is minimized over z_i and $\mathsf{z}_{i,j}$. In this way, both steps can be decoupled over the agents.

Solving problem (8) until convergence requires various ADMM iterations. Incorporating this sequence in step 6 of Algorithm 1 is however not desired in practice as it would involve too high a computation and communication load. Therefore a DMPC scheme is proposed that executes only one ADMM iteration per control update starting from the solution of the previous iteration. The idea is that ADMM converges while the vehicles are heading towards their destination.

Algorithm 2 summarizes the DMPC strategy. The algorithm starts with executing a number of ADMM iterations before the start of the MPC sequence in order to have a good initial trajectory to follow (step 1). In the examples of Section 4, five initial ADMM iterations are used. The steps during an MPC iteration are similar as those listed in Algorithm 1, apart from solving the global optimal control problem. This is replaced by the steps of one ADMM iteration. These steps include the solution of two optimization problems in step 7 and 9 and two communication cycles with the neighbors during step 8 and 11. As an ADMM iteration depends on results computed in a previous iteration, these results are first expressed in the basis corresponding to the new control horizon. This is done in step 6 by computing $\tilde{\mathsf{y}}_i^k$, $\tilde{\mathsf{z}}_i^k$, $\tilde{\mathsf{z}}_{j,i}^k$, $\tilde{\lambda}_i^k$ and $\tilde{\lambda}_{j,i}^k$.

Each agent i now contains two sets of variables: y_i, which are used to drive the vehicle, and z_i and $\mathsf{z}_{i,j}$, representing guesses of respectively y_i and y_j of its neighbors j. Step 7 implies that y_i^k satisfy at all time the local constraints, represented by Y_i, while step 9 ensures that z_i^k and $\mathsf{z}_{i,j}^k$ satisfy the formation constraints. The ADMM iterations let the y and z variables converge towards each other.

4. Examples

The proposed DMPC strategy is illustrated and analyzed by means of three example cases. The first one considers the numerical simulation of a formation of quadrotors flying in a changing environment. The second one considers differential wheeled robots moving in relative formation. The third example validates the DMPC approach experimentally on three robotic platforms. Additional examples are found in the supporting toolbox [27].

Algorithm 2 ADMM based Distributed MPC

```
 1: Perform $n$ ADMM iterations and get $\mathsf{y}_i^0$, $\mathsf{z}_i^0$, $\mathsf{z}_{j,i}^0$, $\lambda_i^0$, $\lambda_{j,i}^0$
 2: **Repeat** every $\Delta T$: $k = 0, 1, \dots$
 3:     Extract $q_i^k(t)$, $u_i^k(t)$ from $\mathsf{y}_i^k$
 4:     Start following trajectories $q_i^k(t)$, $u_i^k(t)$
 5:     Estimate $\hat{q}_i^k = q_i^k((k+1)\Delta T)$, $\hat{u}_i^k = u_i^k((k+1)\Delta T)$
 6:     Update horizon and compute $\tilde{\mathsf{y}}_i^k$, $\tilde{\mathsf{z}}_i^k$, $\tilde{\mathsf{z}}_{j,i}^k$, $\tilde{\lambda}_i^k$, $\tilde{\lambda}_{j,i}^k$
 7:     Compute $\mathsf{y}_i^{k+1}$, using $\hat{q}_i^k$, $\hat{u}_i^k$ as initial conditions:
            $\mathsf{y}_i^{k+1} := \underset{\mathsf{y}_i \in \mathsf{Y}_i(\hat{q}_i^k, \hat{u}_i^k)}{\operatorname{argmin}} \mathcal{L}_{\rho,i}(\mathsf{y}_i, \tilde{\mathsf{z}}_i^k, \tilde{\lambda}_i^k, \mathsf{y}_i, \tilde{\mathsf{z}}_{j,i}^k, \tilde{\lambda}_{j,i}^k)$
 8:     Communication with agent $j$, $\forall j \in \mathcal{N}_i$:
            send $\mathsf{y}_i^{k+1}$, receive $\mathsf{y}_j^{k+1}$
 9:     Compute $\mathsf{z}_i^{k+1}$ and $\mathsf{z}_{i,j}^{k+1}$:
            $\begin{pmatrix} \mathsf{z}_i^{k+1} \\ \mathsf{z}_{i,j}^{k+1} \end{pmatrix} := \begin{array}{ll} \underset{\mathsf{z}_i, \mathsf{z}_{i,j}}{\operatorname{argmin}} & \mathcal{L}_{\rho,i}(\mathsf{y}_i^{k+1}, \mathsf{z}_i, \tilde{\lambda}_i^k, \mathsf{y}_j^{k+1}, \mathsf{z}_{i,j}, \tilde{\lambda}_{i,j}^k) \\ \text{s.t.} & \mathsf{g}_{i,j}(\mathsf{z}_i, \mathsf{z}_{i,j}) = 0 \,, \quad \forall j \in \mathcal{N}_i \end{array}$
10:     Compute $\lambda_i^{k+1}$ and $\lambda_{i,j}^{k+1}$:
            $\lambda_i^{k+1} := \tilde{\lambda}_i^k + \rho(\mathsf{y}_i^{k+1} - \mathsf{z}_i^{k+1})$
            $\lambda_{i,j}^{k+1} := \tilde{\lambda}_{i,j}^k + \rho(\mathsf{y}_j^{k+1} - \mathsf{z}_{i,j}^{k+1}) \,, \quad \forall j \in \mathcal{N}_i$
11:     Communication with agent $j$, $\forall j \in \mathcal{N}_i$:
            send $\mathsf{z}_{i,j}^{k+1}$ and $\lambda_{i,j}^{k+1}$, receive $\mathsf{z}_{j,i}^{k+1}$ and $\lambda_{j,i}^{k+1}$
12: **Until** target reached
```

4.1. Quadrotor formation flight

The first use case considers a formation of two-dimensional quadrotors as described in Section 2.1. This formation should move between two vertical walls in order to reach a desired destination, as is presented in the first snapshot of Figure 3. Using the output trajectories $y_i = [q_{i,1}, q_{i,2}]^T$, the objective of a vehicle i at iteration k is chosen as

$$J_i(y_i) = \int_{\mathcal{T}^k} \|y_i(t) - y_{T,i}\|_1 dt \qquad (11)$$

in order to steer it as fast as possible to its destination $y_{T,i}$. The formation constraints are described in absolute terms meaning that they are expressed with respect to a fixed inertial frame:

$$g_{i,j}(y_i, y_j) = y_i - y_j - \Delta y_{i,j} = 0 \,,$$

where $\Delta y_{i,j}$ represents the desired relative position between a vehicle i and j. The interconnection between the vehicles is circular. This means that each agent has two neighbors, one on the left and one on the right. The nominal control horizon length equals $T = 5\,\text{s}$, the control period $\Delta T = 0.1\,\text{s}$ and nominal width of the polynomial interval $\Delta\xi = 0.5\,\text{s}$. Applied to this example, step 7 of Algorithm 2 is a nonconvex nonlinear program, while step 9

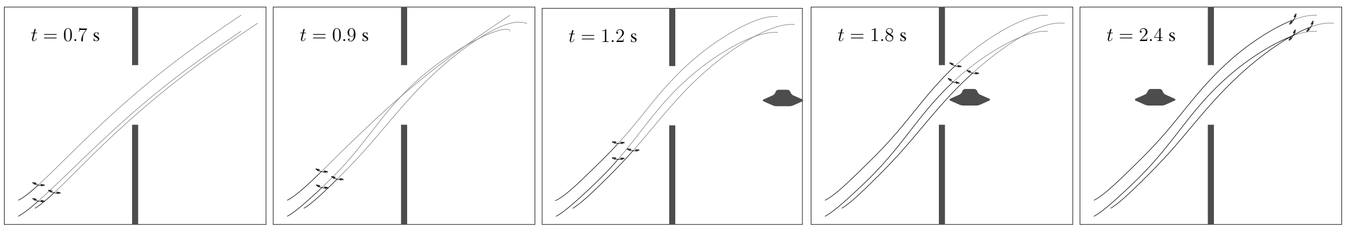


Figure 3: Motion trajectories for a formation of three quadrotors in a dynamic environment. At $t = 0.8$ s, the vehicles observe an obstacle approaching with a constant velocity. The proposed DMPC approach allows to avoid this obstacle while flying in formation.

is a quadratic program with linear equality constraints. The former is solved with Ipopt [35], the latter is solved in one step by directly solving the KKT system. For both problems CasADi [36] is used as symbolic framework, algorithmic differentiation tool and interface to the numerical solvers.

A receding-horizon algorithm is especially beneficial in the presence of disturbances. This is illustrated for a formation of three quadrotors that suddenly encounter a moving object in their airspace. The resulting motion is illustrated in Figure 3. At $t = 0.8$ s, the aerial vehicles observe the object and determine its position and (constant) velocity. This information is used to adapt the trajectories in order to avoid collisions during the considered control horizon. The first adapted trajectories are applied at $t = 0.9$ s. One can observe in Figure 3 that these trajectories contain large formation violations. This is possible as these constraints are implicitly relaxed in the augmented Lagrangian. However, as the updates proceed, these trajectories converge towards a new optimum such that the quadrotors can avoid the object while attaining the formation.

The convergence is monitored in Figure 4 by means of the combined residual c^k and formation error ϵ_{F}^k. The combined residual measures both the primal and the dual residual simultaneously [37] and is formulated as

$$c^k = \sum_{i=1}^{N} \Big(\rho \| \mathsf{y}_i^k - \mathsf{z}_i^k \|_2^2 + \sum_{j \in \mathcal{N}_i} \rho \| \mathsf{y}_j^k - \mathsf{z}_{i,j}^k \|_2^2 + \rho \| \mathsf{z}_i^k - \tilde{\mathsf{z}}_i^{k-1} \|_2^2 + \sum_{j \in \mathcal{N}_i} \rho \| \mathsf{z}_{i,j}^k - \tilde{\mathsf{z}}_{i,j}^{k-1} \|_2^2 \Big) .$$

The formation error is computed as the average relative deviation of a vehicle with respect to the formation center. With $y_{\mathrm{c}}^k(t) = \frac{1}{N} \sum_i y_i^k(t)$ the formation center of the planned trajectories during iteration k and $\Delta y_{i,\mathrm{c}}$ the ideal relative position of a vehicle i with respect to the formation center, this becomes

$$\epsilon_{\mathrm{F}}^k = \frac{1}{|\mathcal{T}^k|} \int_{\mathcal{T}^k} \frac{1}{N} \sum_{i=1}^{N} \frac{\| y_i^k(t) - y_{\mathrm{c}}^k(t) - \Delta y_{i,\mathrm{c}} \|_2}{\| \Delta y_{i,\mathrm{c}} \|_2} dt ,$$

where $\mathcal{T}^k$ indicates the considered control horizon at iteration k and $|\mathcal{T}^k|$ represents its length. At $t = 0.9$ s, a jump occurs in both the residual and formation error due to the change in the environment. From that point, the DMPC approach converges to the new optimum and the formation error is quickly reduced during the subsequent iterations.

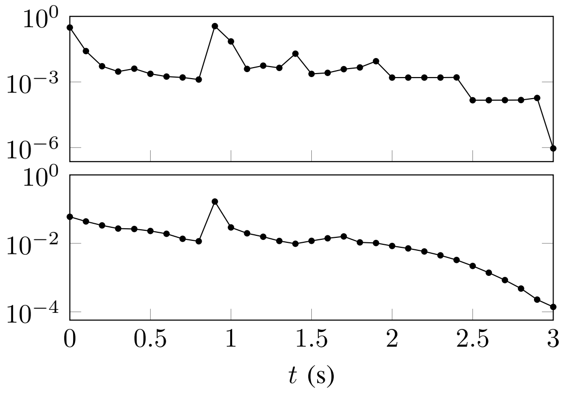


Figure 4: Combined residual c^k and formation error ϵ_{F}^k as a function of time for a formation of three quadrotors in a dynamic environment. At $t = 0.9$ s, a jump occurs in the residual and formation error due to a change in the environment. From that point, the DMPC approach converges to a new optimum.

Solving the quadrotor formation problem with the proposed DMPC approach takes on average 57 ms per control update[1]. As a comparison, solving this problem with a central MPC approach as described in Algorithm 1, takes 187 ms per update.

4.2. Relative formation of differential wheeled robots

In order to further illustrate the versatility of the proposed approach, a different example case is presented. It considers three differential wheeled robots as described in Section 2.1. They should move in a relative formation. This means that the formation is defined with respect to the local robot frames, and acts as if the vehicles were rigidly attached to each other. This is desired in applications as cooperative transportation. Each vehicle expresses in its local robot frame its desired relative position

[1]All simulations are performed on a notebook with Intel Core i5-4300M CPU @ 2.60 GHz x 4 processor and 8 GB of memory.

$\Delta q_{i,\mathrm{c}} = [\Delta x_{i,\mathrm{c}}, \Delta y_{i,\mathrm{c}}]^T$ with respect to the formation center, where $\Delta x_{i,\mathrm{c}}$ and $\Delta y_{i,\mathrm{c}}$ are expressed along respectively the longitudinal and lateral direction. The formation center $q_i^{\mathrm{c}} = \left[q_{i,1}^{\mathrm{c}}, q_{i,2}^{\mathrm{c}}\right]^T$, seen from vehicle i and expressed in the inertial frame becomes

$$\begin{aligned} q_{i,1}^{\mathrm{c}} &= q_{i,1} - \Delta x_{i,\mathrm{c}} \cos q_{i,3} + \Delta y_{i,\mathrm{c}} \sin q_{i,3}\,, \\ q_{i,2}^{\mathrm{c}} &= q_{i,2} - \Delta x_{i,\mathrm{c}} \sin q_{i,3} - \Delta y_{i,\mathrm{c}} \cos q_{i,3}\,. \end{aligned} \tag{12}$$

The variables q_i^{c} are introduced as slack (spline) variables in the optimization problem. After substituting (6), and relaxing the equality constraint, (12) is reformulated as polynomial constraints

$$\begin{aligned} -\epsilon \le (1+z_i^2)(q_{i,1} - q_{i,1}^{\mathrm{c}}) - (1-z_i^2)\Delta x_{i,\mathrm{c}} + 2z_i \Delta y_{i,\mathrm{c}} \le \epsilon\,, \\ -\epsilon \le (1+z_i^2)(q_{i,2} - q_{i,2}^{\mathrm{c}}) - 2z_i \Delta x_{i,\mathrm{c}} - (1-z_i^2)\Delta y_{i,\mathrm{c}} \le \epsilon\,, \end{aligned} \tag{13}$$

where ϵ is a small positive number. The constraint relaxation is necessary to find a feasible solution in the proposed spline parameterization. Inequality constraints (13) are added to the local constraint set Y_i of vehicle i. The formation is implied by enforcing consensus on the formation center, expressed in the inertial frame:

$$g_{i,j}(q_i^{\mathrm{c}}, q_j^{\mathrm{c}}) = q_i^{\mathrm{c}} - q_j^{\mathrm{c}} = 0\,.$$

Using this approach, step 9 of Algorithm 2 is again reduced to a linear equality constrained quadratic program.

For this example, the proposed DMPC approach is applied and the resulting point-to-point motion and tangential velocity $u_{i,1}$ for each vehicle are presented in Figure 5. The robots form a rigid triangle that can translate and rotate in order to reach the destination in an obstructed environment. Figure 5b illustrates how an outer lying vehicle is constrained by its maximum velocity of 1 m/s while covering the S-turn. As a similar objective as (11) is used, this proves the optimality of the solution. Note that the gradual increase and decrease in velocity at the beginning and the end is due to the spline parameterization. The average required time to perform a control update equals 392 ms. Currently ways of reducing this update time are investigated.

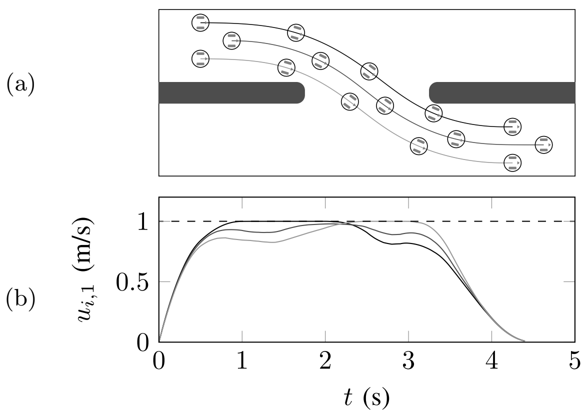


Figure 5: Three differential wheeled robots moving in relative formation. (a) illustrates the the motion, while (b) represents the corresponding tangential velocity.

4.3. Experimental validation on holonomic platforms

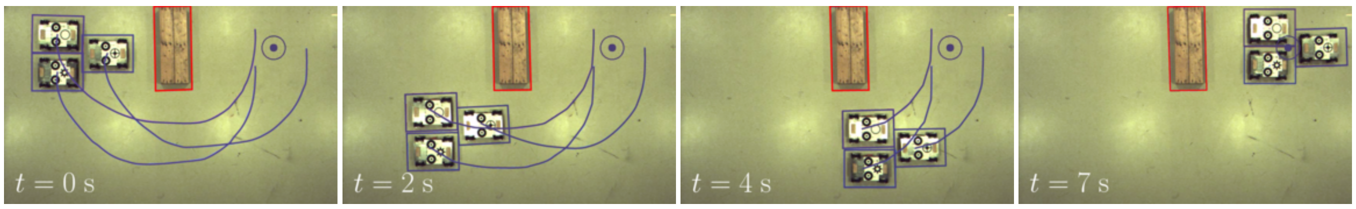
The presented approach is implemented and demonstrated on three in-house developed robotic platforms as illustrated in Figure 6. Each platform is equipped with four independently driven Mecanum wheels, which renders the system holonomic. The platform contains an Odroid XU4 single board computer on which the DMPC algorithm is implemented. The platforms' hardware communicate with each other over Wi-Fi. The setup also includes a ceiling camera that detects the platforms' absolute position and orientation. Each robot merges this information with its local encoder measurements for retrieving a fast and accurate state estimate.

The DMPC iterations are performed periodically with a rate of 4 Hz. The bottleneck for increasing this rate is solving the non-linear program of step 7 in Algorithm 2 by Ipopt on the Odroid XU4. The DMPC algorithm generates state and input trajectories that are provided as reference to a low-level feedback controller running at 100 Hz.

Figure 7 presents two point-to-point tasks performed on the robotic platforms illustrated by snapshots taken from the ceiling camera. In the first example a triangular formation should avoid a static obstacle in the environment. The target position for the formation center is indicated by the concentric gray circles. The gray lines indicate the predicted position trajectories for each platform. The rectangles indicate the shapes used for avoiding collisions between vehicles and obstacle. To account for disturbances and model-plant mismatch it is desired to stay further away from obstacles than strictly necessary for avoiding collisions. Therefore a similar construction as described in [26] is used to motivate the vehicles to move an extra distance of 5 cm away from the obstacle. The second example considers a formation of two vehicles. The platform located in the center of the field, indicated by the red rectangle, acts as an obstacle which is steered manually with a gamepad. At $t = 1$ s it suddenly starts to move downwards. The other vehicles are using an estimate of the obstacle's position and velocity in order to predict its motion. In this way they can adapt their trajectories to pass the vehicle from above in stead of trying to move underneath the vehicle, which was the initial plan. The DMPC algorithm makes sure the two vehicles attain their formation. A video illustrating the experiments is found in the `readme` file of the supporting toolbox [27].

4.4. OMG-tools

The proposed DMPC approach is implemented as part of a more general spline-based motion control toolbox with the name Optimal Motion Generation-tools [27]. This toolbox is written in Python and uses CasADi [36] as symbolic framework and interface to solvers. The goal of OMG-tools is to facilitate the modeling, simulation and


(a)

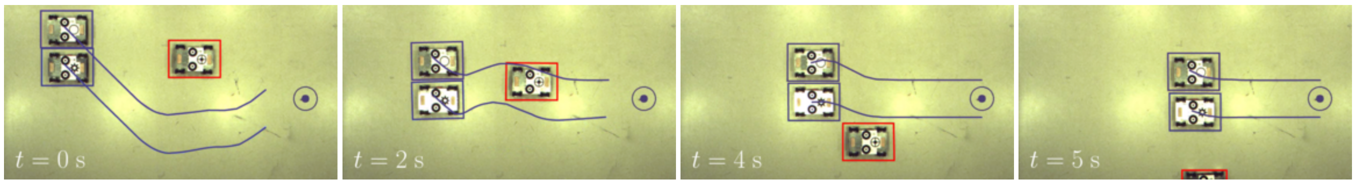

(b)

Figure 7: Experimental validation of the DMPC approach on a formation of robotic platforms moving (a) in a static environment and (b) in a dynamic environment.

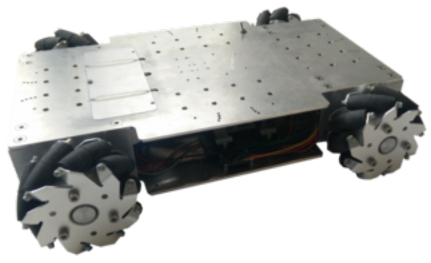


Figure 6: In-house developed robotic platform

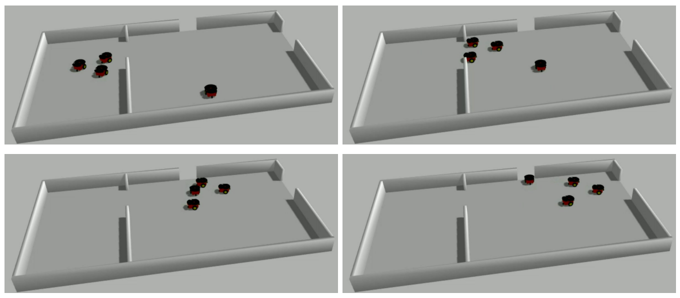


Figure 8: Simulation example performed with ROS and Gazebo illustrating a formation of three Pioneer P3-DX robots avoiding a fourth independent P3-DX.

deployment of spline-based motion problems. The software is easily coupled with existing widely used robotic frameworks such as ROS [38]. An example considering a formation of three Pioneer P3-DX robots moving in formation and avoiding a fourth independently driven P3-DX is presented in Figure 8 . In order to generate this example, OMG-tools is integrated in ROS and coupled to the robotic simulator Gazebo [39]. A movie demonstrating the example is found in the `readme` file of OMG-tools.

Apart from all examples presented in this paper, the toolbox provides more illustrations of the capabilities of the presented method. For a quick overview, the interested reader is referred to the `readme` file that contains various illustrative animations and videos of experiments.

5. Conclusion

This paper presents a novel DMPC strategy for controlling multi-vehicle systems moving in formation. In contrast to existing approaches we allow to handle realistic problems, including nonlinear vehicle dynamics and obstacle avoidance constraints. In order to retrieve an efficient algorithm, efforts are made to reduce the computational and communicational burden. On the one hand the size of the overall optimization is reduced by using a spline parameterization for the vehicles' trajectories. On the other hand, the computational load of this problem is distributed over the vehicles by using the Alternating Direction Method of Multipliers (ADMM). A DMPC scheme is proposed where one ADMM iteration is performed during each control update. Numerical simulations with nonlinear vehicle dynamics and an experimental validation on holonomic platforms show that the ADMM iterations converge sufficiently fast for the considered multi-vehicle application. The proposed DMPC approach and various illustrative examples are implemented as part of a spline-based motion control toolbox.

Appendix A. Collision avoidance constraints

Collision avoidance constraints between a vehicle and an obstacle are constructed by imposing the existence of a separating hyperplane between their shapes. Note that this construction can only separate convex shapes [40]. Figure A.9 illustrates this approach. Suppose the vehicle's shape is represented by a circle with center v and radius r while the obstacle is a convex polygon with vertices w_i. Demanding the separation of both shapes by a

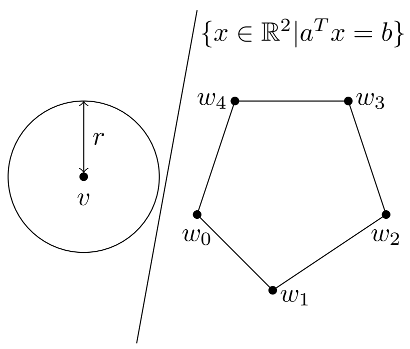


Figure A.9: Separating a circle and a convex polygon by a separating line.

line $\{x \in \mathbb{R}^2 | a^T x = b\}$ is achieved with the following constraints:

$$\begin{aligned} -a^T v + b &\geq r \|a\|_2 \,, \\ a^T w_i - b &\geq 0 \,, \quad \forall i \in \{0, \dots, 4\} \,. \end{aligned}$$

The center v of the circle depends on the position of the vehicle and therefore on the output trajectory y as discussed in Section 2.1. In order to avoid collisions at all time, the separating line is allowed to change over time. Both $a(\cdot)$ and $b(\cdot)$ are introduced as time dependent optimization variables and are parameterized as splines. In the event of a dynamic environment the vertices $w_i(\cdot)$ of the obstacle can be parameterized as (known) spline trajectories in order to incorporate predictions concerning the obstacle's motion. After reformulating the constraints to be differentiable, they are written as

$$\begin{aligned} -a(t)^T v(y(t)) + b(t) &\geq r \,, \\ a(t)^T w_i(t) - b(t) &\geq 0 \,, \quad \forall i \in \{0, \dots, 4\} \\ a(t)^T a(t) &\leq 1 \,, \\ &\forall t \in [0, T] \,. \end{aligned} \tag{A.1}$$

When $v(\cdot)$ is a polynomial function of y, its derivatives and anti-derivatives, constraints (A.1) are polynomial in the spline variables $y(t)$, $a(t)$ and $b(t)$ such that the constraint enforcement of Section 3.1 can be used to reformulate them in a finite set of constraints.

References

[1] M. Beukenberg, D. Hummel, Aerodynamics, performance and control of airplanes in formation flight, in: ICAS, Congress, 17 th, Stockholm, Sweden, Proceedings., Vol. 2, 1990, pp. 1777–1794.

[2] J. P. Desai, J. P. Ostrowski, V. Kumar, Modeling and control of formations of nonholonomic mobile robots, Robotics and Automation, IEEE Transactions on 17 (6) (2001) 905–908.

[3] S. Mastellone, D. M. Stipanović, C. R. Graunke, K. A. Intlekofer, M. W. Spong, Formation control and collision avoidance for multi-agent non-holonomic systems: Theory and experiments, The International Journal of Robotics Research 27 (1) (2008) 107–126.

[4] M. A. Lewis, K.-H. Tan, High precision formation control of mobile robots using virtual structures, Autonomous Robots 4 (4) (1997) 387–403.

[5] R. W. Beard, J. Lawton, F. Y. Hadaegh, A coordination architecture for spacecraft formation control, IEEE Transactions on control systems technology 9 (6) (2001) 777–790.

[6] T. Balch, R. C. Arkin, Behavior-based formation control for multirobot teams, Robotics and Automation, IEEE Transactions on 14 (6) (1998) 926–939.

[7] R. Olfati-Saber, Flocking for multi-agent dynamic systems: Algorithms and theory, IEEE Transactions on automatic control 51 (3) (2006) 401–420.

[8] K. Hengster-Movrić, S. Bogdan, I. Draganjac, Multi-agent formation control based on bell-shaped potential functions, Journal of intelligent & robotic systems 58 (2) (2010) 165–189.

[9] W. Dong, Robust formation control of multiple wheeled mobile robots, Journal of Intelligent & Robotic Systems 62 (3) (2011) 547–565.

[10] F. Xiao, L. Wang, J. Chen, Y. Gao, Finite-time formation control for multi-agent systems, Automatica 45 (11) (2009) 2605–2611.

[11] J. Ghommam, H. Mehrjerdi, M. Saad, F. Mnif, Formation path following control of unicycle-type mobile robots, Robotics and Autonomous Systems 58 (5) (2010) 727–736.

[12] M. Saska, V. Vonásek, T. Krajník, L. Přeučil, Coordination and navigation of heterogeneous mav–ugv formations localized by a hawk-eye-like approach under a model predictive control scheme, The International Journal of Robotics Research 33 (10) (2014) 1393–1412.

[13] P. D. Christofides, R. Scattolini, D. M. de la Peña, J. Liu, Distributed model predictive control: A tutorial review and future research directions, Computers & Chemical Engineering 51 (2013) 21–41.

[14] R. R. Negenborn, J. Maestre, Distributed model predictive control: An overview and roadmap of future research opportunities, Control Systems, IEEE 34 (4) (2014) 87–97.

[15] J. M. Maestre, R. R. Negenborn, Distributed model predictive control made easy, Springer, 2014.

[16] J. Maestre, M. Ridao, A. Kozma, C. Savorgnan, M. Diehl, M. Doan, A. Sadowska, T. Keviczky, B. De Schutter, H. Scheu, et al., A comparison of distributed MPC schemes on a hydropower plant benchmark, Optimal Control Applications and Methods 36 (3) (2015) 306–332.

[17] P. Trodden, A. Richards, Distributed model predictive control of linear systems with persistent disturbances, International Journal of Control 83 (8) (2010) 1653–1663.

[18] M. Farina, A. Perizzato, R. Scattolini, Application of distributed predictive control to motion and coordination problems for unicycle autonomous robots, Robotics and Autonomous Systems 72 (2015) 248–260.

[19] R. L. Raffard, C. J. Tomlin, S. P. Boyd, Distributed optimization for cooperative agents: Application to formation flight, in: Proceedings of the 2004 Conference on Decision and Control, Vol. 3, 2004, pp. 2453–2459.

[20] A. J. Häusler, A. Saccon, A. P. Aguiar, J. Hauser, A. M. Pascoal, Energy-optimal motion planning for multiple robotic vehicles with collision avoidance, IEEE Transactions on Control Systems Technology 24 (3) (2016) 867–883.

[21] F. Farokhi, I. Shames, K. H. Johansson, Distributed MPC via dual decomposition and alternative direction method of multipliers, in: Distributed Model Predictive Control Made Easy, Springer, 2014, pp. 115–131.

[22] H. Y. Ong, J. Gerdes, Cooperative collision avoidance via proximal message passing, in: Proceedings of the 2015 American Control Conference, 2015, pp. 4124–4130.

[23] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends® in Machine Learning 3 (1) (2011) 1–122.

[24] W. Van Loock, G. Pipeleers, J. Swevers, B-spline parameterized optimal motion trajectories for robotic systems with guaranteed

constraint satisfaction, Mechanical Sciences 6 (2) (2015) 163–171.

[25] T. Mercy, W. Van Loock, G. Pipeleers, Real-time motion planning in the presence of moving obstacles, in: Proceedings of the 2016 European Control Conference, 2016, pp. 1586–1591.

[26] R. Van Parys, G. Pipeleers, Spline-based motion planning in an obstructed 3D environment, in: Proceedings of the 20th IFAC World Congress, 2017, pp. 8998–9003.

[27] R. Van Parys, T. Mercy, OMG-tools, `https://github.com/meco-group/omg-tools` (2016).

[28] R. Van Parys, G. Pipeleers, Online distributed motion planning for multi-vehicle systems, in: Proceedings of the 2016 European Control Conference, 2016, pp. 1580–1585.

[29] C. R. Hargraves, S. W. Paris, Direct trajectory optimization using nonlinear programming and collocation, Journal of Guidance, Control, and Dynamics 10 (4) (1987) 338–342.

[30] N. Petit, M. B. Milam, R. M. Murray, Inversion based constrained trajectory optimization, in: 5th IFAC symposium on nonlinear control systems, Vol. 5, Citeseer, 2001.

[31] P. Martin, R. M. Murray, P. Rouchon, Flat systems, equivalence and trajectory generation, Tech. rep., California Institute of Technology (2003).

[32] M. Fliess, J. Lévine, P. Martin, P. Rouchon, Flatness and defect of non-linear systems: introductory theory and examples, International Journal of Control 61 (6) (1995) 1327–1361.

[33] C. De Boor, A practical guide to splines, revised edition, vol. 27 of applied mathematical sciences (2001).

[34] R. Findeisen, F. Allgöwer, Computational delay in nonlinear model predictive control, in: Proc. Int. Symp. Adv. Contr. of Chem. Proc, 2004, pp. 427–432.

[35] A. Wächter, L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Mathematical programming 106 (1) (2006) 25–57.

[36] J. Andersson, A general-purpose software framework for dynamic optimization, Ph.D. thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium (2013).

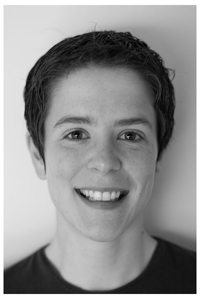
[37] T. Goldstein, B. O'Donoghue, S. Setzer, R. Baraniuk, Fast alternating direction optimization methods, SIAM Journal on Imaging Sciences 7 (3) (2014) 1588–1623.

[38] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, Ros: an open-source robot operating system, in: ICRA workshop on open source software, Vol. 3, Kobe, 2009, p. 5.

[39] N. Koenig, A. Howard, Design and use paradigms for gazebo, an open-source multi-robot simulator, in: Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, Vol. 3, IEEE, 2004, pp. 2149–2154.

[40] S. Boyd, L. Vandenberghe, Convex optimization, Cambridge university press, 2004.

Ruben Van Parys received the M.Sc. degree in industrial science, in 2011, from the Katholieke Hogeschool Brugge-Oostende (KHBO), Belgium, and the M.Sc degree in mechanical engineering, in 2014, from the Katholieke Universiteit Leuven (KU Leuven), Belgium, where he is currently working towards the Ph.D. degree. His research interests include optimal motion control of mechatronic systems with a focus on cooperating multi-vehicle systems.



Goele Pipeleers is an assistant professor at the Department of Mechanical Engineering of the KU Leuven. She received her M.Sc. degree in mechanical engineering and her Ph.D. degree in mechanical engineering from the KU Leuven, in 2004 and 2009, respectively. She has been a Post-doctoral Fellow of the Research Foundation Flanders, and a visiting scholar at the Colorado School of Mines and at the University of California Los Angeles. Her research interests include convex optimization, optimal and robust control, and their applications in mechatronics.