Distributed model predictive formation control with inter-vehicle collision avoidance*

Ruben Van Parys[1] and Goele Pipeleers[1]

Abstract— **This work presents a distributed model predictive control strategy for generating collision-free point-to-point motions for a formation of multiple vehicles. It extends a synchronous distributed model predictive control (DMPC) scheme presented in earlier work that allows violations of the formation requirement when necessary, for example to avoid suddenly appearing obstacles or to move through narrow passages. In order to obtain safe motion trajectories during these violations, the current paper presents an approach for including inter-vehicle collision avoidance constraints in the DMPC scheme. The collision avoidance is achieved by separating each pair of vehicles by a shared hyperplane and two methods are proposed for updating the separating hyperplane over the DMPC cycles. Simulations with formations of holonomic vehicles demonstrate the capability of the presented strategy.**

I. Introduction

Boosted by advances in computational power and communication technologies, formation control of networked multi-vehicle systems has gained increasing interest over the last decades. Examples of application areas include surveillance, search and rescue, distributed sensing and cooperative transportation. Although it is desired that the vehicles move according to a specified formation configuration, in some cases it is beneficial to allow violations on these formation requirements. This for example to avoid suddenly appearing obstacles or to move through narrow passages. When the vehicles leave the formation they might come in close contact to one another and therefore it is of major importance to avoid collisions between them. This work focuses on the inclusion of inter-vehicle collision avoidance constraints in a previous presented distributed formation control approach [1].

Many of the state-of-the-art distributed collision avoidance approaches are based on the concept of velocity obstacles, or a generalization hereof [2]–[4]. In these methods the vehicles decide upon a constant input profile for a certain time horizon that is as close as possible to their desired inputs but also guarantees collision avoidance during the horizon. The lack of considering a free motion trajectory in the prediction horizon is a major drawback that reduces the optimality of the resulting motions.

In contrast, a distributed model predictive control (DMPC) strategy allows to compute a free trajectory in the control horizon. The inclusion of inter-vehicle collision avoidance constraints in a DMPC scheme requires that each vehicle knows about the planned intentions of the other vehicles in order to avoid them. Depending on how this is achieved, three general types of DMPC schemes are distinguished: sequential, iterative and synchronous DMPC.

In *sequential* DMPC the vehicles solve their control problems one after another [5]–[7]. In this way the posterior vehicle uses communicated information of the anterior ones in order to avoid them. This however implies a priorization of the vehicles' objectives based on the sequence order. In addition, these strategies often lead to low control rates as the sequential solving of each vehicle's control problem should happen during one MPC update.

An *iterative* DMPC scheme uses an iterative distributed optimization approach to solve the multi-vehicle optimal control problem until convergence during each MPC cycle [8], [9]. Each iteration requires the solution of an optimization problem and involves substantial communication between the vehicles, which typically results in low MPC update rates.

In *synchronous* DMPC, each vehicle solves its optimal control problem once simultaneously in each MPC period. The inclusion of inter-vehicle collision avoidance in a synchronous DMPC scheme is however not straightforward. Because the vehicles compute their planned trajectories in parallel, every vehicle lacks the information on the intentions of its neighbors in order to avoid them properly. This problem is addressed in [10], [11] by including a compatibility constraint that limits the distance between a newly planned motion trajectory and the trajectory computed during the previous MPC cycle. Each vehicle has received the previous planned trajectories of the others and knows the allowed maximum deviation from them during the current update. In this way a vehicle can compute a non-colliding motion trajectory. The approach has as disadvantage that it uses the distance to the closest neighbor for determining the compatibility constraint. This construction is rather conservative and can deteriorate the control performance.

This paper presents a DMPC approach for generating collision-free point-to-point motions for a formation of multiple vehicles. It extends the synchronous DMPC scheme presented in [1], which provides each vehicle with its own distinct motion controller while inter-vehicle communica-

*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 from 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.

[1]The authors are with the MECO Research Group, Department of Mechanical Engineering, Division PMA, KU Leuven, 3001 Leuven, Belgium. ruben.vanparys@kuleuven.be

[1]Member of Flanders Make.

tion allows the vehicles to cooperate in order to move in formation. The scheme allows the vehicles to violate formation constraints when necessary. In order to obtain safe motion trajectories during these violations, the current paper investigates possibilities for including inter-vehicle collision avoidance constraints in this distributed control scheme. The collision avoidance is achieved by separating each pair of vehicles by a shared hyperplane and two methods are proposed for updating this separating hyperplane over the DMPC cycles. The result is a novel synchronous DMPC scheme that outperforms the existing approaches of [10], [11] as the separating hyperplanes allow a larger space for the vehicles to optimize their trajectories. The current paper focuses on holonomic vehicles moving in a two dimensional plane. The approach can be extended for non-linear and three dimensional spaces using ideas from [12]. These extensions do not influence the presented ideas for distributed inter-vehicle collision avoidance.

Section II introduces the considered multi-vehicle formation problem. Section III gives a short overview of the DMPC scheme presented in [1], while Section IV explains how inter-vehicle collision avoidance is introduced in this scheme. The proposed approach is illustrated with numerical examples in Section V. Finally, Section VI draws concluding remarks.

II. Problem formulation

The problems considered in this work search for position trajectories $x_i(\cdot)$ for N different velocity steered holonomic vehicles i in order to bring them from an initial position to a terminal position. Optimal trajectories are obtained by minimizing the sum of all vehicles' objectives J_i. In order to steer a vehicle as close as possible to its destination $x_{T,i}$ during the control horizon $[0,\, T]$, a vehicle's objective is formulated as

$$J_i(x_i) = \int_0^T \|x_i(t) - x_{T,i}\|_1 dt\,.$$

A vehicle's constraints are divided in a local and a global set. The local constraints are expressed as $x_i(\cdot) \in \mathcal{X}_i$ and comprise input constraints, expressed as limitations on a vehicle's velocity,

$$\dot{x}_{\min,i} \leq \dot{x}_i \leq \dot{x}_{\max,i}\,,$$

and collision avoidance constraints with obstacles in the environment. These are constructed by imposing the existence of a separating hyperplane between the vehicle's and obstacle's shape. Suppose the geometry of vehicle i is approximated as a circle with radius r_i, while the obstacle is represented by a circle with midpoint x_o and radius r_o. Demanding the separation of both circles by a hyperplane $\{x \in \mathbb{R}^2 | a^T x = b\}$ is achieved with the following set of constraints:

$$\begin{aligned} -a(t)^T x_i(t) + b(t) &\geq r_i\,, \\ a(t)^T x_o(t) - b(t) &\geq r_o\,, \\ a(t)^T a(t) &\leq 1\,, \\ \forall t \in [0,\, T]\,. \end{aligned} \tag{1}$$

As a vehicle's position varies over time and possible obstacles move around, the separating hyperplane, and therefore a and b, is allowed to change over time.

Global constraints ensure the formation preservation and inter-vehicle collision avoidance. Formation constraints are imposed by fixing the relation between the positions x_i and x_j of respectively a vehicle i and its neighbor j using a desired relative position $\Delta x_{i,j}$:

$$g_{i,j}(x_i(t), x_j(t)) = x_i(t) - x_j(t) - \Delta x_{i,j} = 0\,, \forall t \in [0,\, T]. \tag{2}$$

The set of neighbors that vehicle i shares constraint (2) with is represented by $\mathcal{N}_i^f$. Inter-vehicle collision avoidance constraints $h_{i,j}$ are formulated similar as (1) with separating hyperplane parameterized by time-dependent $a_{i,j}(\cdot)$ and $b_{i,j}(\cdot)$. These constraints are imposed between each pair of vehicles (i,j). The related neighbor set is therefore chosen as $\mathcal{N}_i^c = \{1, \ldots, N\} \setminus \{i\}$.

The resulting optimization problem is summerized as:

$$\begin{aligned} \underset{\substack{x_i(\cdot),\, a_{i,j}(\cdot),\, b_{i,j}(\cdot),\\ i=1,\ldots,N,\, j \in \mathcal{N}_i^c}}{\text{minimize}} \quad & \sum_{i=1}^{N} J_i(x_i) \\ \text{subject to} \quad & x_i(\cdot) \in \mathcal{X}_i\,, \\ & g_{i,j}(x_i(t), x_j(t)) = 0,\, \forall j \in \mathcal{N}_i^f\,, \\ & h_{i,j}(x_i(t), x_j(t), a_{i,j}(t), b_{i,j}(t)) \leq 0\,, \\ & \forall j \in \mathcal{N}_i^c\,, \\ & \forall t \in [0, T], \quad \forall i \in \{1, \ldots, N\}\,. \end{aligned} \tag{3}$$

III. Spline-based DMPC

This section briefly recapitulates the DMPC scheme presented in [1]. First, problem (3) is translated into a nonlinear program by adopting a spline parameterization for the position trajectories. Second, an approach is presented for solving the resulting problem in an efficient distributed and receding-horizon fashion.

A. Spline parameterization

Problem (3) is infinite dimensional, comprising both infinitely many optimization variables and constraints, as the optimization variables are functions and they must satisfy the constraints at all time instants. To cope with the infinitely many optimization variables, the trajectory variables are approximated as piecewise polynomials and are parameterized in a B-spline basis [13]:

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

with B-spline basis $b = [b_1, \ldots, b_n]^T$ and B-spline coefficients $\mathsf{x}_i = [\mathsf{x}_{i,1}, \ldots, \mathsf{x}_{i,n}]^T$. 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:

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

Because (anti-)derivatives and polynomial functions of a spline are splines as well, also polynomial constraints on spline trajectories and their (anti-)derivatives can be handled in the same way. For holonomic vehicles all constraints are naturally polynomial in the motion trajectories x_i and their derivatives. For details on how to account for nonholonomic vehicles, the reader is referred to [12], [14]. Finally by using a spline parameterization for $x_i(\cdot)$, $a_{i,j}(\cdot)$ and $b_{i,j}(\cdot)$ and the constraint replacement, problem (3) is formulated in terms of the spline coefficients x_i, $\mathsf{a}_{i,j}$ and $\mathsf{b}_{i,j}$:

$$\begin{aligned} \underset{\substack{\mathsf{x}_i,\,\mathsf{a}_{i,j},\,\mathsf{b}_{i,j},\\ i=1,\dots,N,\,j\in\mathcal{N}_i^c}}{\text{minimize}} \quad & \sum_{i=1}^{N} \mathsf{J}_i(\mathsf{x}_i) \\ \text{subject to} \quad & \mathsf{x}_i \in \mathsf{X}_i\,, \\ & \mathsf{g}_{i,j}(\mathsf{x}_i,\mathsf{x}_j) = 0,\, \forall j \in \mathcal{N}_i^f\,, \\ & \mathsf{h}_{i,j}(\mathsf{x}_i,\mathsf{x}_j,\mathsf{a}_{i,j},\mathsf{b}_{i,j}) \le 0\,, \forall j \in \mathcal{N}_i^c\,, \\ & \forall i \in \{1,\dots,N\}\,. \end{aligned} \tag{4}$$

The objective and constraint functions, reformulated in terms of spline coefficients, are written in Sans-serif font.

B. ADMM-based Distributed MPC

In problem (4), the local constraints only apply to one specific vehicle while the objective is composed of the vehicles' individual objectives. The only coupling is imposed by the formation constraints $\mathsf{g}_{i,j}$ and inter-vehicle collision avoidance constraints $\mathsf{h}_{i,j}$. This subsection describes a control scheme that decouples the problem in order to distribute the computational load of solving the control problem among the vehicles. As an intermediate step in deriving an overall DMPC scheme, this subsection presents the decoupling of the formation constraints. The inter-vehicle collision avoidance constraints are temporarily omitted and will be added to the DMPC scheme in Section IV. The decoupling is derived using the Alternating Direction Method of Multipliers (ADMM) [15]. Before applying ADMM, problem (4) is reformulated as the following equivalent one:

$$\begin{aligned} \underset{\mathsf{x}_i,\,\mathsf{z}_i,\,\mathsf{z}_{i,j},\ i=1,\dots,N}{\text{minimize}} \quad & \sum_{i=1}^{N} \mathsf{J}_i(\mathsf{x}_i) \\ \text{subject to} \quad & \mathsf{x}_i \in \mathsf{X}_i \\ & \mathsf{g}_{i,j}(\mathsf{z}_i,\mathsf{z}_{i,j}) = 0\,, \quad \forall j \in \mathcal{N}_i^f \\ & \mathsf{x}_i = \mathsf{z}_i\,,\, \mathsf{x}_j = \mathsf{z}_{i,j}\,, \quad \forall j \in \mathcal{N}_i^f \\ & \forall i \in \{1,\dots,N\}\,, \end{aligned} \tag{5}$$

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 x_i and the neighbor's trajectory x_j. The idea behind ADMM is to solve a dual problem of (5), which is obtained by dualizing equality constraints $\mathsf{x}_i = \mathsf{z}_i$ and $\mathsf{x}_j = \mathsf{z}_{i,j}$. ADMM utilizes the augmented Lagrangian function which adds an extra quadratic penalty term in order to improve robustness and to yield milder convergence assumptions. For problem (5) this function is

$$\begin{aligned} \mathcal{L}_\rho = \sum_{i=1}^{N} \Big(& \mathsf{J}_i(\mathsf{x}_i) + \lambda_i^T(\mathsf{x}_i - \mathsf{z}_i) + \frac{\rho}{2}\|\mathsf{x}_i - \mathsf{z}_i\|_2^2 \\ & + \sum_{j\in\mathcal{N}_i} \big(\lambda_{i,j}^T(\mathsf{x}_j - \mathsf{z}_{i,j}) + \frac{\rho}{2}\|\mathsf{x}_j - \mathsf{z}_{i,j}\|_2^2\big)\Big)\,, \\ = \sum_{i=1}^{N} & \mathcal{L}_{\rho,i}(\mathsf{x}_i,\mathsf{z}_i,\lambda_i,\mathsf{x}_j,\mathsf{z}_{i,j},\lambda_{i,j})\,, \end{aligned} \tag{6}$$

where λ_i and $\lambda_{i,j}$ represent the dual variables associated with the dualized constraints. ADMM uses an iterative gradient ascent method to find the dual optimum. In each iteration the evaluation of the dual function is split in two consecutive steps where first the Lagrangian function (6) is minimized over x_i and second over z_i and $\mathsf{z}_{i,j}$. In this way these two steps and the gradient update of λ_i and $\lambda_{i,j}$ can be performed fully decoupled.

Solving problem (5) until convergence requires various ADMM iterations. Incorporating this sequence in one MPC cycle is however not desired in practice as it would involve too high a computation and communication load. Therefore [1] proposes a DMPC scheme that executes only one ADMM iteration per control update starting from the solution of the previous iterations. The idea is that ADMM converges while the vehicles are heading towards their destination.

Algorithm 1 summarizes the DMPC strategy. For more details, the reader is referred to [1]. Important to capture is that each vehicle i contains two sets of variables: x_i, which are used to drive the vehicle, and z_i and $\mathsf{z}_{i,j}$, representing guesses of respectively x_i and x_j of its neighbors j. In (7) a

Algorithm 1 ADMM based Distributed MPC

1: Initial ADMM iterations to get x_i^0, z_i^0, $\mathsf{z}_{j,i}^0$, λ_i^0, $\lambda_{j,i}^0$
2: **Repeat** every ΔT: $k = 0, 1, \dots$
3: Start following trajectory $x_i^k(t)$
4: Estimate $\hat{x}_i^k = x_i^k((k+1)\Delta T)$
5: Update horizon and transform x_i^k, z_i^k, $\mathsf{z}_{j,i}^k$, λ_i^k, $\lambda_{j,i}^k$
6: Compute x_i^{k+1}, using $\hat{x}_i^k$ as initial conditions:

$$\mathsf{x}_i^{k+1} := \underset{\mathsf{x}_i\in\mathsf{X}_i(\hat{x}_i^k)}{\operatorname{argmin}}\ \mathcal{L}_{\rho,i}(\mathsf{x}_i,\mathsf{z}_i^k,\lambda_i^k,\mathsf{x}_i,\mathsf{z}_{j,i}^k,\lambda_{j,i}^k) \tag{7}$$

7: Communication with agent $j\,, \forall j \in \mathcal{N}_i^f$:
send x_i^{k+1}, receive x_j^{k+1}
8: Compute z_i^{k+1} and $\mathsf{z}_{i,j}^{k+1}$:

$$\begin{pmatrix}\mathsf{z}_i^{k+1}\\ \mathsf{z}_{i,j}^{k+1}\end{pmatrix} := \begin{aligned}\underset{\mathsf{z}_i,\mathsf{z}_{i,j}}{\operatorname{argmin}} \quad & \mathcal{L}_{\rho,i}(\mathsf{x}_i^{k+1},\mathsf{z}_i,\lambda_i^k,\mathsf{x}_j^{k+1},\mathsf{z}_{i,j},\lambda_{i,j}^k) \\ \text{s.t.} \quad & \mathsf{g}_{i,j}(\mathsf{z}_i,\mathsf{z}_{i,j}) = 0\,, \quad \forall j \in \mathcal{N}_i^f\end{aligned} \tag{8}$$

9: Compute λ_i^{k+1} and $\lambda_{i,j}^{k+1}$:

$$\begin{aligned} \lambda_i^{k+1} &:= \lambda_i^k + \rho(\mathsf{x}_i^{k+1} - \mathsf{z}_i^{k+1}) \\ \lambda_{i,j}^{k+1} &:= \lambda_{i,j}^k + \rho(\mathsf{x}_j^{k+1} - \mathsf{z}_{i,j}^{k+1})\,, \quad \forall j \in \mathcal{N}_i^f \end{aligned} \tag{9}$$

10: Communication with agent $j\,, \forall j \in \mathcal{N}_i^f$:
send $\mathsf{z}_{i,j}^{k+1}$ and $\lambda_{i,j}^{k+1}$, receive $\mathsf{z}_{j,i}^{k+1}$ and $\lambda_{j,i}^{k+1}$
11: **Until** target reached

vehicle i solves a local optimal control problem, ensuring that x_i^k always satisfies the local constraints, represented by X_i. Problem (8) ensures that z_i^k and $\mathsf{z}_{i,j}^k$ satisfy the formation constraints. The ADMM iterations let the x and z variables converge towards each other. Because x variables are used to drive the vehicle and because equality constraints between x and z are dualized in the optimal control problem (7), the formation constraints are imposed in a soft manner. This allows the vehicles to violate the formation requirement especially when no other option is available. In order to guarantee safe motions, it is important to include collision avoidance constraints between the vehicles.

IV. Inter-vehicle collision avoidance

This section describes how inter-vehicle collision avoidance requirements are included in the formation control problem. As described in Section II, collision avoidance between two vehicles i and j is achieved by separating them with a hyperplane. This hyperplane can change over time and is parameterized by a set of spline variables $a_{i,j}(\cdot)$ and $b_{i,j}(\cdot)$, shared by the two vehicles. In order to include inter-vehicle collision avoidance in the DMPC scheme of Algorithm 1, an appropriate decoupling should be implemented.

A dual decomposition approach similar as ADMM does however not work in this case. Such strategy provides each vehicle with a local copy of their separating hyperplane and dualizes equality constraints on these local versions. As in the DMPC scheme of Algorithm 1 only one iteration per MPC cycle is executed, no guarantee is given that the trajectories computed in a particular MPC cycle are non-colliding.

Instead the inter-vehicle collision avoidance is decoupled in the primal space. Vehicle i and j keep the evolution of their separating hyperplane, described by $a_{i,j}(\cdot)$ and $b_{i,j}(\cdot)$, fixed while solving their local optimal control problem (7) and constraint their position trajectories to stay within one of the moving halfspaces created by the separating hyperplane. Separating hyperplanes are imposed between each pair of vehicles such that a vehicle's position is constrained to stay withing a time-varying intersection of halfspaces. With $a_{i,j}^k(\cdot)$ and $b_{i,j}^k(\cdot)$ the fixed separating hyperplane parameters during iteration k, and using the convention that $a_{i,j} = -a_{j,i}$ and $b_{i,j} = -b_{j,i}$, a vehicle i introduces the following constraints:

$$\begin{aligned} -a_{i,j}^k(t)^T x_i(t) + b_{i,j}^k(t) &\geq r_i s_{i,j}(t)\,, \\ a_{i,j}^k(t)^T a_{i,j}^k(t) &\leq s_{i,j}(t)^2\,, \\ s_{i,j} &\geq 0\,, \\ \forall j \in \mathcal{N}_i^c\,, \forall t \in [0,\, T]\,, \end{aligned} \tag{10}$$

where $s_{i,j}(\cdot)$ are slack spline variables that represent an upper bound on $\|a_{i,j}^k(\cdot)\|_2$. Constraints (10) are reformulated using the constraint replacement described in Section III-A and are added to the local constraint set X_i of the local optimal control problem (7).

The separating hyperplanes should be updated properly over the MPC cycles. As the introduction of constraints (10) with fixed $a_{i,j}^k$ and $b_{i,j}^k$ can drastically reduce the feasible set of the local optimal control problems, it is important to guarantee recursive feasibility over the MPC updates. This means that $a_{i,j}^{k+1}$ and $b_{i,j}^{k+1}$ should be chosen such that at least one feasible solution exists for the optimal control problems of cycle $k+1$. In addition it is desired to choose the separating hyperplanes such that the feasible set of the vehicle's optimal control problems is as large as possible, giving sufficient space to optimize new trajectories. Taking the aforementioned requirements in mind, two methods are proposed for updating the separating hyperplanes.

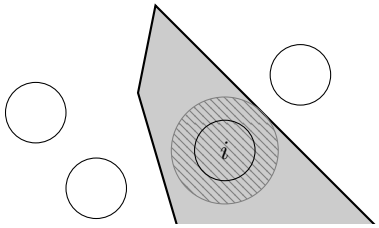

Fig. 1: Determining new separating hyperplanes at a given time in the prediction horizon for vehicle i using Method A.

1) Method A: The first approach uses the position trajectories $x_i^k(\cdot)$ computed during cycle k to determine the separating hyperplane parameters $a_{i,j}^{k+1}(\cdot)$ and $b_{i,j}^{k+1}(\cdot)$ used in cycle $k+1$. At each time instant the new separating hyperplane is chosen to lie in the middle of two vehicles, perpendicular to the line connecting them:

$$\begin{aligned} a_{i,j}^{k+1}(\cdot) &= x_j^k(\cdot) - x_i^k(\cdot)\,, \\ b_{i,j}^{k+1}(\cdot) &= \frac{1}{2}(x_j^k(\cdot) - x_i^k(\cdot))^T(x_j^k(\cdot) + x_i^k(\cdot))\,. \end{aligned}$$

In this way, at each time instant the space is equally divided between the two vehicles to optimize their new trajectories. Figure 1 illustrates the hyperplanes separating a vehicle i with its three neighbors at a particular time instant t. The hyperplanes create the gray surface wherein the vehicle's circle shape is allowed when computing a new position $x_i^{k+1}(t)$ for time t. Note how method A guarantees recursive feasibility as previous computed trajectories are always a feasible solution for the optimal control problem of the next MPC cycle.

2) Method B: The second approach updates the previous hyperplane parameters using the sensitivities of the optimal objective function of a vehicle's local optimal control problem to the hyperplane parameters. With $\mathcal{L}_{\rho,i}^{k*}$ the optimal objective value of a vehicle i's optimal control problem (7) during MPC cycle k, the coefficients of the separating hyperplane parameters are updated using a gradient step:

$$\mathsf{a}_{i,j}^{k+1} = \mathsf{a}_{i,j}^k - \alpha_{i,j}^k\left(\frac{d\mathcal{L}_{\rho,i}^{k*}}{d\mathsf{a}_{i,j}^k} - \frac{d\mathcal{L}_{\rho,j}^{k*}}{d\mathsf{a}_{j,i}^k}\right),$$

where sensitivities $\frac{d\mathcal{L}_{\rho,i}^{k*}}{d\mathsf{a}_{i,j}}$ are computed using optimal dual variables of (7) [16]. The sensitivities express how vehicles i and j want to adapt the shared separating hyperplane in order to improve their local objectives. Both sensitivities are taken equally into account for updating the separating hyperplane. The step size $\alpha_{i,j}^k$ is determined such that recursive feasibility

Algorithm 2 DMPC with inter-vehicle collision avoidance

1: Get initial x_i^0, $\mathsf{a}_{i,j}^0$, $\mathsf{b}_{i,j}^0$, z_i^0, $\mathsf{z}_{j,i}^0$, λ_i^0, $\lambda_{j,i}^0$
2: **Repeat** every ΔT: $k = 0, 1, \ldots$
3: Start following trajectory $x_i^k(t)$
4: Estimate $\hat{x}_i^k = x_i^k((k+1)\Delta T)$
5: Update horizon and transform x_i^k, $\mathsf{a}_{i,j}^k$, $\mathsf{b}_{i,j}^k$, z_i^k, $\mathsf{z}_{j,i}^k$, λ_i^k, $\lambda_{j,i}^k$
6: Compute x_i^{k+1} using (7) including constraints (10) with $\mathsf{a}_{i,j}^k$ and $\mathsf{b}_{i,j}^k$
7: Communication with agent $j\,, \forall j \in \mathcal{N}_i^f \cup \mathcal{N}_i^c$: send x_i^{k+1} ($\frac{d\mathcal{L}_{\rho,i}^{k*}}{d\mathsf{a}_{i,j}^k}$), receive x_j^{k+1} ($\frac{d\mathcal{L}_{\rho,j}^{k*}}{d\mathsf{a}_{j,i}^k}$)
8: Compute z_i^{k+1} and $\mathsf{z}_{i,j}^{k+1}$ using (8)
9: Compute λ_i^{k+1} and $\lambda_{i,j}^{k+1}$ using (9)
10: Compute $\mathsf{a}_{i,j}^{k+1}$ and $\mathsf{b}_{i,j}^{k+1}$ using method A or B
11: Communication with agent $j\,, \forall j \in \mathcal{N}_i^f$: 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

is guaranteed. A line search on $\alpha_{i,j}^k$ is performed such that previous trajectories $x_i^k(\cdot)$ and $x_j^k(\cdot)$ allow a separation by the updated hyperplane. The parameters $\mathsf{b}_{i,j}^{k+1}$ are not determined using sensitivity information but are chosen such that the hyperplane crosses the point in the middle of the line connecting the two vehicles. Experiments have revealed that using sensitivity information for updating $\mathsf{b}_{i,j}^k$ gives poor results.

Algorithm 2 presents the resulting scheme when incorporating method A or B in the steps of Algorithm 1. Method A requires a vehicle to know the computed trajectories of its neighbors. This information is available as it was already provided by the communication during step 7. Method B also requires this information in order to perform the line search on $\alpha_{i,j}^k$ and needs in addition to communicate a neighbor's sensitivity information (step 7). Both method A and B assume that previous sets of separating hyperplanes allow feasible motion trajectories. This implies that the DMPC algorithm should start from feasible hyperplanes $\mathsf{a}_{i,j}^0$ and $\mathsf{b}_{i,j}^0$. These can be generated for instance using a sequential approach. Each vehicle solves a point-to-point problem sequentially. Posterior vehicles use communicated position trajectories of anterior ones to determine a collision-free trajectory x_i^0 and separating hyperplanes $\mathsf{a}_{i,j}^0$, $\mathsf{b}_{i,j}^0$. Any priorization imposed by this sequential scheme is eliminated by the further steps of Algorithm 2.

Note how the use of separating hyperplanes allows a much larger space to reoptimize trajectories when compared to the distance restriction used in [10], [11]. This is illustrated in Figure 1. While method A allows a new position $x_i^{k+1}(t)$ to lie in the gray surface constructed by the separating lines, the method of [10], [11] restricts $x_i^{k+1}(t)$ to lie in the hatched circle.

V. Numerical results

This section illustrates the proposed DMPC scheme with two collision critical formation maneuvers. Animations of both examples are uploaded on the readme of a spline-based motion planning toolbox[1] developed by the authors.

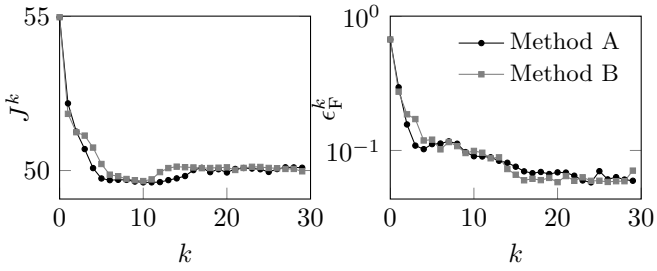

Fig. 2: Comparison of method A and B based on the convergence of the objective J^k and formation error ϵ_{F}^k.

A. Formation moving through a narrow passage

A first example considers a formation of three vehicles that should move through a narrow passage to reach the destination. This example is used to compare the two methods for updating the separating hyperplanes described in Section IV. The comparison is made by monitoring the convergence of the distributed method of Algorithm 2 when employing it to iteratively solve an optimal control problem. This means that the control horizon and the initial conditions used in (7) are not updated over the iterations. The convergence is monitored in Figure 2 by means of the total objective value $J^k = \sum_{i=1}^N J_i(x_i^k)$ and the formation error ϵ_{F}^k. The latter is computed as the average relative deviation of a vehicle with respect to the formation center. With $x_c^k(t) = \frac{1}{N}\sum_i x_i^k(t)$ the formation center of the planned trajectories during iteration k and $\Delta x_{i,c}$ the ideal relative position of a vehicle i with respect to the formation center, this becomes

$$\epsilon_{\mathrm{F}}^k = \frac{1}{T}\int_0^T \frac{1}{N}\sum_{i=1}^N \frac{\|x_i^k(t) - x_{\mathrm{c}}^k(t) - \Delta x_{i,\mathrm{c}}\|_2}{\|\Delta x_{i,\mathrm{c}}\|_2} dt\,.$$

As seen in Figure 2 both methods have a similar convergence. Also from other examples it was noticed that no method is clearly outperforming the other. Based on performance the choice between both methods is hard to make. However as method B requires additional communicated information and needs to execute a line search, which can be computational demanding, method A is the preferred choice from practical viewpoint. Figure 3 demonstrates the formation's motion after performing 30 iterations. In the snapshot at $t = 3.5$ s the allowed regions, defined by the separating hyperplanes, are illustrated for each vehicle.

B. Formation in a dynamic environment

Using an MPC algorithm is especially beneficial in the presence of disturbances. This is illustrated in Figure 4 for a formation of five vehicles that suddenly encounter two moving obstacles. At $t = 1.1$ s and $t = 1.3$ s the vehicles

[1] https://github.com/meco-group/omg-tools

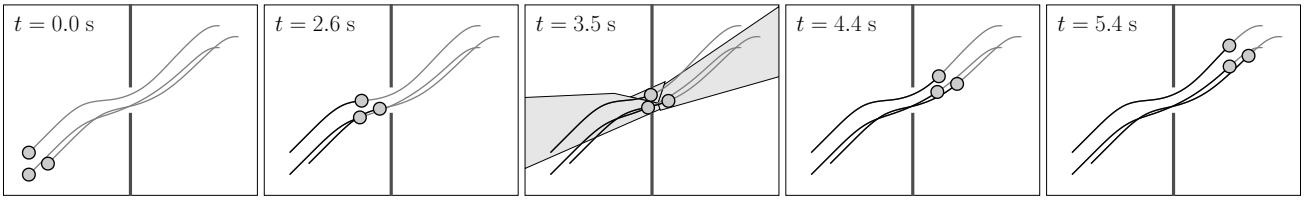

Fig. 3: Motion trajectories for a formation of three vehicles moving through a narrow passage. The gray lines indicate predicted trajectories while black lines represent covered trajectories. The gray surfaces in the third snapshot illustrate the allowed region for each vehicle at $t = 3.5$ s defined by the separating hyperplanes.

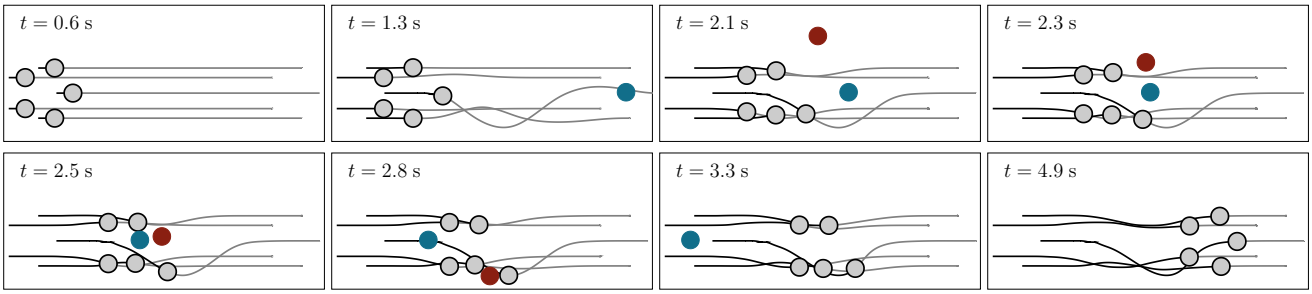

Fig. 4: Motion trajectories for a formation of five vehicles in a dynamic environment. At $t = 1.1$ s and $t = 1.3$ s the vehicles observe respectively the blue and red object. The proposed approach allows the vehicles to adapt their trajectories to avoid the obstacles. Adapted trajectories violate the formation requirements but remain collision-free.

observe respectively the blue and red object and determine its position and (constant) velocity. This information is taken into account in order to avoid them. Adapted trajectories violate the formation requirements but remain collision-free.

VI. Conclusion

This paper presents a DMPC approach for steering a formation of vehicles with guaranteed inter-vehicle collision avoidance. It extends a previously presented DMPC approach that decouples the formation constraints based on the Alternating Direction Method of Multipliers (ADMM) and that only executes one ADMM iteration per MPC cycle. This allows the vehicles to violate the formation constraints when necessary and requires the inclusion of inter-vehicle anti-collision constraints to guarantee safe motions. These constraints are based on the separation of each vehicle pair by a shared hyperplane and are decoupled in the primal space. Two methods are proposed for updating the shared hyperplane over the MPC cycles and numerical simulations show that both methods allow similar performance. The possibilities of the presented approach are demonstrated by numerical examples with formations of holonomic vehicles moving through static and dynamic environments.

References

[1] R. Van Parys and G. Pipeleers, "Distributed MPC for multi-vehicle systems moving in formation," *Robotics and Autonomous Systems*, vol. 97, pp. 144–152, 2017.

[2] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey, "Clearpath: highly parallel collision avoidance for multi-agent simulation," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 177–187, ACM, 2009.

[3] J. Van Den Berg, S. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," *Robotics research*, pp. 3–19, 2011.

[4] D. Bareiss and J. van den Berg, "Generalized reciprocal collision avoidance," *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1501–1514, 2015.

[5] Y. Kuwata, A. Richards, T. Schouwenaars, and J. P. How, "Distributed robust receding horizon control for multivehicle guidance," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 4, pp. 627–641, 2007.

[6] G. Chaloulos, P. Hokayem, and J. Lygeros, "Distributed hierarchical mpc for conflict resolution in air traffic control," in *American Control Conference (ACC), 2010*, pp. 3945–3950, IEEE, 2010.

[7] F. Tedesco, D. M. Raimondo, A. Casavola, and J. Lygeros, "Distributed collision avoidance for interacting vehicles: a command governor approach," *IFAC Proceedings Volumes*, vol. 43, no. 19, pp. 293–298, 2010.

[8] Y. Kuwata and J. P. How, "Cooperative distributed robust trajectory optimization using receding horizon MILP," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 2, pp. 423–431, 2011.

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

[10] P. Wang and B. Ding, "A synthesis approach of distributed model predictive control for homogeneous multi-agent system with collision avoidance," *International Journal of Control*, vol. 87, no. 1, pp. 52–63, 2014.

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

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

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

[14] W. Van Loock, G. Pipeleers, and J. Swevers, "B-spline parameterized optimal motion trajectories for robotic systems with guaranteed constraint satisfaction," *Mechanical Sciences*, vol. 6, no. 2, pp. 163–171, 2015.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[16] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.