2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Coordination of Multiple Vessels Via Distributed Nonlinear Model Predictive Control

L. Ferranti*, R. R. Negenborn*, T. Keviczky* and J. Alonso-Mora*

***Abstract*— This work presents a method for multi-robot trajectory planning and coordination based on nonlinear model predictive control (NMPC). In contrast to centralized approaches, we consider the distributed case where each robot has an on-board computation unit to solve a local NMPC problem and can communicate with other robots in its neighborhood. We show that, thanks to tailored interactions (i.e., interactions designed according to a nonconvex alternating direction method of multipliers, or ADMM, scheme), the proposed solution is equivalent to solving the centralized control problem. With some communication exchange, required by the ADMM scheme at given synchronization steps, the safety of the robots is preserved, that is, collisions with neighboring robots are avoided and the robots stay within the bounds of the environment. In this work, we tested the proposed method to coordinate three autonomous vessels at a canal intersection. Nevertheless, the proposed approach is general and can be applied to different applications and robot models.**

I. INTRODUCTION

One of the challenges to ensure safe navigation of autonomous vehicles (such as cars or vessels) in urban environments is that of generating safe trajectories that coordinate with other traffic participants. A typical scenario is that of intersections, where efficient navigation can be achieved with tight coordination between the interacting participants. This is a multi-robot motion planning problem, for which we present a distributed method based on Nonlinear Model Predictive Control (NMPC).

Common approaches for intersection control rely on the computation of controlled invariant sets and scheduling [1], [2], [3], [4], but typically require a discretization and allocation of the environment and limit their results to systems with order-preserving dynamics. In contrast, our approach operates directly in continuous space via a constrained optimization and applies to arbitrary dynamics. Several authors have proposed centralized optimization-based approaches for multi-robot coordination and intersection negotiation [3], [5], [6]. In contrast, our approach does not require a central coordinator and is decentralized via a communication channel.

In recent years, MPC has gained attention in applications with fast dynamics, such as automotive [7], [3], [8], waterborne transport [9], [10] and aerospace [11] thanks to great improvements in terms of solvers used for online optimization [12], [13], [14], [15]. We build on [8], which relies on the model predictive contouring control (MPCC) formulation proposed in [16], [17], to design our model predictive controller. Compared to [8], we extend the problem formulation to a network of agents. We also rely on the fast solver Forces Pro [15] in our implementation to solve the local optimization problems that arise from our proposed decomposition.

Our contribution is a decentralized approach for multi-robot trajectory planning based on NMPC, which does not require a central coordinator nor a discretization of the environment. In particular, we first formulate the navigation problem as if a central coordinator was available. Then, we show how to reformulate this *centralized problem* in a distributed way without the need of a central coordinator. We achieve this goal by introducing *(i)* a new set of decision variables that act as shared variables among the different robots and *(ii)* a new set of constraints that handle the *consensus* among the robots. The resulting optimization problem can be decomposed among the robots, which work in parallel, thanks to the use of an operator-splitting technique, namely a modified version of the alternating direction method of multipliers (ADMM) suitable for nonconvex optimization [18]. We evaluate our method for the control of autonomous vessels at a canal intersection. Yet, the method is general and can be applied to other multi-robot systems and scenarios.

In the remainder of the paper, all the vectors are indicated with a bold symbol. The 2-norm of a vector $\mathbf{u}$ is $\|\mathbf{u}\|$.

II. PRELIMINARIES

A. Model of the agents

In the remainder of the paper we consider autonomous agents whose dynamics can be represented by the following nonlinear discrete-time model:

$$\boldsymbol{x}_i(t+1) = f_i(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t)), i \in \mathcal{I}_V, \tag{1}$$

where $\boldsymbol{x}_i(t) \in \mathbb{R}^{n_i}$ represents the state of agent i, $\boldsymbol{u}_i(t) \in \mathbb{R}^{m_i}$ represents the associated control command, $f_i : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \to \mathbb{R}^{n_i}$ represents the (possibly) nonlinear dynamics of agent i, and $\mathcal{I}_V := \{1, \dots, V\}$.

The model description above is general and can be employed in different applications, ranging from automotive to aerospace. In case of a continuous-time systems, the model above can be obtained by discretization of the continuous-time dynamics, as in the example provided in Section V.

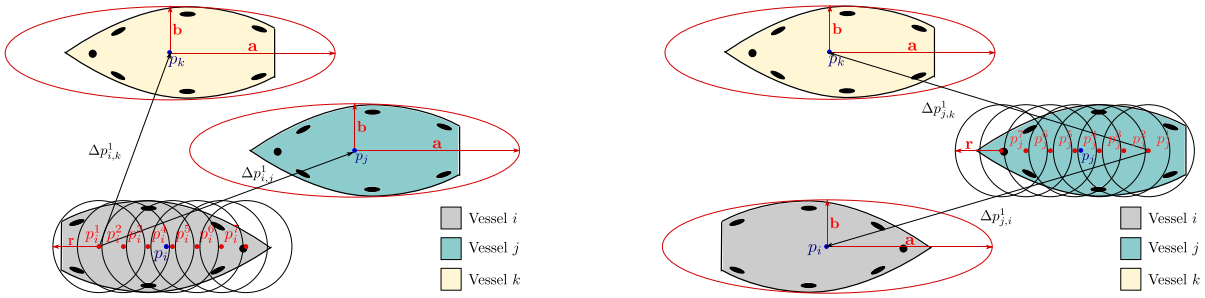
B. Collision avoidance constraints

The proposed control design should be able to avoid collisions among different agents. Hence, in the following,

J. Alonso-Mora is supported by the NWO Veni n. 15916.

*The authors are with the Faculty of Mechanical, Maritime, and Materials Engineering, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands {l.ferranti, r.r.negenborn, t.keviczky, j.alonsomora}@tudelft.nl

The authors thank W. Schwarting for sharing the ICRA17 paper's code.
Video available at https://youtu.be/wJMTmLLwmDo



(a) Perspective of agent i.

(b) Perspective of agent j.

Fig. 1: Agents' representation for collision avoidance: the circles represent the agent, while the ellipses describe its neighbors.

we introduce the notation and the strategy used to represent each agent and formulate the collision avoidance constraints in the control problem. In this work, we use a strategy similar to the one proposed in [8] for autonomous cars. Other strategies exist to formulate the collision avoidance constraints [19], [20]. Compared to [19], we do not linearize the collision avoidance constraints, while compared to [20], we do not rely on scheduling. Compared to [8], the agents can exchange information concerning their position.

Without loss of generality, we consider two agents i and j. We note that the method applies in an straight-forward way to the case of $V \geq 2$ agents by including additional constraints for each of the other agents analogously to the case for agent j (as depicted in Figure 1 where we added a third agent, namely agent k, to represent a more general scenario). Figure 1 depicts the proposed approach from the perspective of agent i (Figure 1a) and agent j (Figure 1b), respectively. Consider Figure 1a. Agent i is represented as n_{disc} discs centered in $\boldsymbol{p}_i^h$ (where we used $\boldsymbol{p}$ to indicate the position on the (x, y) plane in the body frame and n_{disc} is the number of discs used to describe the agent), $h \in \mathcal{I}^{\text{disc}} := \{1, 2, \ldots, n_{\text{disc}}\}$. From the perspective of agent i, agent j is represented as an ellipse with semi-major axis a (longitudinal direction) and b (lateral direction), respectively. This description of the agents' shape is well suited for NMPC and not overly conservative.

To formulate the collision avoidance constraints for agent i, we define the distance between the center of the h-th disc—used to represent agent i—and the center of the ellipses—used to define agent j from agent i point of view—as follows (and depicted in Figure 1a):

$$\Delta \boldsymbol{p}_{i,j}^h = \boldsymbol{p}_i^h - \boldsymbol{p}_j, h \in \mathcal{I}^{\text{disc}}. \tag{2}$$

Hence, the collision avoidance constraints from agent i at time t are[1]:

$$\underbrace{\left(R(\eta_j)\Delta \boldsymbol{p}_{i,j}^h\right)^{\mathrm{T}} \begin{bmatrix} \frac{1}{a^2+r^2} & 0 \\ 0 & \frac{1}{b^2+r^2} \end{bmatrix} R(\eta_j)\Delta \boldsymbol{p}_{i,j}^h}_{\boldsymbol{c}_{i,j}^h} > 1, h \in \mathcal{I}^{\text{disc}}, \tag{3}$$

where $R(\eta_j)$ is the rotation matrix related to agent j with orientation η_j. Note the constraints above require $\boldsymbol{p}_j$ and $\boldsymbol{\eta}_j$. We can repeat this argument from the perspective of agent j, as depicted in Figure 1b. In particular, from the perspective of agent j, the information concerning $\boldsymbol{p}_i$ and η_i is needed.

III. CENTRALIZED NMPC

Our approach relies on MPC. The MPC controller recursively solves an optimization problem based on the available plant measurements to compute the optimal sequence of control commands over a finite time window, called prediction horizon. Only the first control command of this sequence is applied to the plant in closed loop in the receding-horizon fashion.

Following [8], we do not rely on a classical reference tracking MPC formulation in which the controller aims to minimize the error with respect to a time-dependent reference signal. The goal of the proposed controller instead is to minimize (for each agent) the following cost[2]:

$$J^e := \boldsymbol{e}^{\mathrm{T}} Q_e \boldsymbol{e} - v_x, \tag{4}$$

where $\boldsymbol{e} \in \mathbb{R}^2$ is the error with respect to a given path $\boldsymbol{p}^{\text{path}}(\phi) \in \mathbb{R}^2$ (where ϕ is the path parameter and v_x is the velocity in the longitudinal direction (17)) in the path's tangential and normal directions. This error is defined as follows:

$$\boldsymbol{e} := \left[e^l \, e^c\right]^{\mathrm{T}}, \tag{5}$$

where e^l and e^c are the *longitudinal error* (i.e., the error in the path's tangential direction) and the *contouring error* (i.e., the error in the path's normal direction), respectively. In particular, at time step t, the longitudinal error is defined as follows:

$$e_t^l := -\left[\cos\bar{\theta}\left(\phi_t\right) \, \sin\bar{\theta}\left(\phi_t\right)\right]\left(\boldsymbol{p}_t - \boldsymbol{p}_t^{\text{ref}}\right), \tag{6}$$

where $\bar{\theta}\left(\phi_t\right)$, $\boldsymbol{p}_t$, and $\boldsymbol{p}_t^{\text{ref}}$ are the heading of the path, the position on the (x, y)-plane of the agent, and the reference path on the (x, y)-plane, respectively. Similarly, at time step t, the contouring error is defined as follows:

$$e_t^c := \left[\sin\bar{\theta}\left(\phi_t\right) \, -\cos\bar{\theta}\left(\phi_t\right)\right]\left(\boldsymbol{p}_t - \boldsymbol{p}_t^{\text{ref}}\right). \tag{7}$$

[1] We assume the same size for all the agents (same a, b, and r in order simplify the notation).

[2] We omit the time and agent dependency when it is clear from the context to simplify the notation.

More details on the derivation of the longitudinal and contouring errors can be found in [8].

The centralized NMPC problem is the following:

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{\phi}} \sum_{i=1}^{V}\sum_{k=1}^{N_i} J_i(\boldsymbol{x}_i(t+k),\boldsymbol{u}_i(t+k),\boldsymbol{\phi}_i(t+k),\bar{\boldsymbol{p}}(t+k)) \quad (8a)$$

$$\text{s. t.}: \boldsymbol{x}_i(t+k+1) = f_i(\boldsymbol{x}_i(t+k), \boldsymbol{u}_i(t+k)), \quad (8b)$$

$$\boldsymbol{\phi}_i(t+k+1) = \boldsymbol{\phi}_i(t+k) + v_{x_i}(t+k)\Delta t_k \quad (8c)$$

$$\boldsymbol{x}_i(t) = \boldsymbol{x}_i^{\text{init}}, \quad (8d)$$

$$G_i(\boldsymbol{x}_i(t+k), \boldsymbol{u}_i(t+k)) \leq \boldsymbol{g}_i, \quad (8e)$$

$$\boldsymbol{c}_{i,j}^h(t+k) > 1, j \neq i, h \in \mathcal{I}^{\text{disc}} \quad (8f)$$

where $\boldsymbol{x}_i, \boldsymbol{u}_i$, and $\boldsymbol{\phi}_i$ represent the predicted evolution of the state, control command, and approximated progress along the path of agent i, respectively, over the prediction horizon N_i. In the remainder of the paper, we define $\boldsymbol{z}_i := \left[\boldsymbol{x}_i^{\text{T}}, \boldsymbol{u}_i^{\text{T}}, \boldsymbol{\phi}_i^{\text{T}}\right]^{\text{T}}$.

For each agent, the goal is to minimize the cost J_i. This cost penalizes the deviation from the reference path of each agent (i.e., the longitudinal and contouring errors (6)-(7)) and maximizes the longitudinal speed. In particular, J_i is defined as follows:

$$J_i := J_i^e - v_{x_i}^{\text{T}} Q^{v_{x_i}} v_{x_i}, \quad (9)$$

where J_i^e penalizes the error with respect to the path for agent i and $Q^{v_{x_i}}$ is the matrix weights the longitudinal speed v_{x_i}. The navigation must comply to *(i)* the dynamics of each agent (expressed by the *dynamic constraints* (8b)) and *(ii)* the physical constraints on the state and control command of each agent (expressed by the constraints (8e)). Furthermore, the navigation must comply to safety requirements of collision avoidance with the other agents moving in the same area (expressed by the nonconvex constraints (8f) detailed in Section II-B).

Solving Problem (8) has some practical limitations. In particular, Problem (8) requires a central node to compute the appropriate control command for all the agents in the network (this node can be required for example to handle an intersection). The central coordinator has to solve the predictive control problem online (i.e., within the sampling time of the fastest agent). This can cause problems for the scalability of the proposed approach, when the number of agents increases. Furthermore, having a central node means that all the agents must be willing to share information concerning their dynamics, constraints, and objectives with the central node. This might be problematic for car or boat manufacturers, which might not be open to share information concerning their products.

IV. DISTRIBUTED NMPC

Consider the centralized problem above (Problem (8)). Note that the only coupling among the different agents is represented by the collision avoidance constraints (8f). Recall that our goal is to solve Problem (8) in a distributed way, that is, without the need of a central coordinator. In the following, first Section IV-A describes our proposed distributed NMPC formulation. Second, Section IV-B shows how to decompose the problem to solve it using a nonconvex version of ADMM proposed in [18].

A. Formulation

We have to modify the centralized problem formulation above to solve it in a distributed way. Each agent needs a local copy of the predicted position and orientation along the prediction horizon of its neighboring agents to solve its local optimization problem. A simple strategy could be that each agent computes its control command using the predicted position and orientation that its neighbors computed at the previous time instant. We could use this strategy to communicate the orientation of the agent to the neighbors, given that the orientation is subject to small variations over time. This strategy, however, is problematic for the (predicted) position on the (x, y) plane of the agent given that it can significantly vary between two time instants. Hence, we can adopt the following strategy to handle the coupling caused by the position and orientation. We introduce the following relationships:

$$\Delta \boldsymbol{p}_{i,j}^h := \boldsymbol{p}_i^h - \bar{\boldsymbol{p}}_j = \bar{\boldsymbol{p}}_i^h - \boldsymbol{p}_j, \bar{\eta}_i := \eta_i, \quad (10)$$

where $\bar{\boldsymbol{p}}_j$ is the local information that agent i has of agent j and $\bar{\boldsymbol{p}}_i$ is the local information that agent j has of agent i. Similarly, $\bar{\eta}_i$ is the local information that agent j has concerning the orientation of agent i. Hence, we reformulate Problem (8) as follows:

$$\min_{\boldsymbol{z}} \sum_{i=1}^{V}\sum_{k=1}^{N_i} J_i(\boldsymbol{z}_i(t+k)) \quad (11a)$$

$$\text{s. t.}: \text{(8b)-(8e)}, \quad (11b)$$

$$\bar{\boldsymbol{p}}_j(t+k) = \boldsymbol{p}_i^h(t+k) - \Delta \boldsymbol{p}_{i,j}^h(t+k),\ j \neq i,\ h \in \mathcal{I}^{\text{disc}} \quad (11c)$$

$$\boldsymbol{p}_j(t+k) = \bar{\boldsymbol{p}}_i^h(t+k) - \Delta \boldsymbol{p}_{i,j}^h(t+k),\ j \neq i,\ h \in \mathcal{I}^{\text{disc}} \quad (11d)$$

$$\boldsymbol{p}_i^h(t+k) = R^h(\boldsymbol{z}_i(t+k))\boldsymbol{z}_i(t+k),\ h \in \mathcal{I}^{\text{disc}} \quad (11e)$$

$$\eta_i(t+k) - \bar{\eta}_i(t+k) = 0 \quad (11f)$$

$$\bar{\boldsymbol{c}}_{i,j}^h(t+k) > 1,\ j \neq i,\ h \in \mathcal{I}^{\text{disc}} \quad (11g)$$

$$\bar{\boldsymbol{c}}_{j,i}^h(t+k) > 1,\ j \neq i,\ h \in \mathcal{I}^{\text{disc}} \quad (11h)$$

In particular, we introduce Constraints (11c)-(11f) to break up the coupling introduced by the collision avoidance constraints and Constraint (11e) to indicate the nonlinear relationship (highlighted by the matrix $R^h(\boldsymbol{z}_i(t+k))$) between the center of the discs describing the agent and the center of the agent. Furthermore, we modify the notation ($\bar{c}_{i,j}^h(t+k)$ instead of $c_{i,j}^h(t+k)$) for the collision avoidance constraints in (3) to indicate that $\bar{\boldsymbol{c}}_{i,j}^h(t+k)$ uses $\bar{\boldsymbol{p}}_j$ instead of $\boldsymbol{p}_j$ ($\bar{\eta}_j$ instead of η_j) and that $\bar{\boldsymbol{c}}_{j,i}^h(t+k)$ uses $\bar{\boldsymbol{p}}_i$ instead of $\boldsymbol{p}_i$. Note that if both (11c) and (11d) are satisfied it means that $\bar{\boldsymbol{p}}_i = \boldsymbol{p}_i$ and $\bar{\boldsymbol{p}}_j = \boldsymbol{p}_j$. Hence, Problem (11) is a reformulation of Problem (8). The difference between the two problems is that Problem (11) can be solved in a distributed fashion by relying on the use of splitting techniques, such as the Alternating Direction Method of Multipliers (ADMM) described below.

B. Problem Decomposition

The approach proposed in this paper strongly relies on ADMM (the interested reader can refer to [21] for a overview for convex optimization). ADMM is a state-of-the-art algorithm used to handle equality constraints in optimization problems such as the ones that can arise from model predictive control applications. In the following, we explain the steps for ADMM.

First, consider the local problem solved by agent i:

$$\min_{\boldsymbol{z}_i, \Delta\boldsymbol{p}^h_{i,j}, \bar{\eta}_j, \boldsymbol{p}^h_i} \sum_{k=1}^{N} J(\boldsymbol{z}(t+k)) \tag{12a}$$

$$\text{s.t.: (8b)-(8e)}, \tag{12b}$$

$$\boldsymbol{p}^h_i(t+k) = R^h(\boldsymbol{z}_i(t+k))\boldsymbol{z}_i(t+k), h \in \mathcal{I}^{\text{disc}} \tag{12c}$$

$$\bar{\boldsymbol{p}}_j(t+k) = \boldsymbol{p}^h_i(t+k) - \Delta\boldsymbol{p}^h_{ij}(t+k), h \in \mathcal{I}^{\text{disc}} \tag{12d}$$

$$\boldsymbol{p}_i(t+k) = \bar{\boldsymbol{p}}^h_j(t+k) - \Delta\boldsymbol{p}^h_{j,i}(t+k), h \in \mathcal{I}^{\text{disc}} \tag{12e}$$

$$\bar{\boldsymbol{c}}^h_{i,j}(t+k) > 1,\ j \neq i,\ h \in \mathcal{I}^{\text{disc}} \tag{12f}$$

$$\bar{\boldsymbol{c}}^h_{j,i}(t+k) > 1,\ j \neq i,\ h \in \mathcal{I}^{\text{disc}} \tag{12g}$$

It is evident from the problem formulation above that $\boldsymbol{z}_i$ and $\boldsymbol{p}^h_i$ are local variables (i.e., a variable whose value is computed on board of agent i). In the remainder of the paper we use $\boldsymbol{\xi}_i := [\boldsymbol{z}_i^{\mathrm{T}}\, \boldsymbol{p}_i^{1^{\mathrm{T}}} \dots \boldsymbol{p}_i^{h^{\mathrm{T}}}]^{\mathrm{T}}$ as the vector of local variables. Vectors $\Delta\boldsymbol{p}^h_{i,j}$, $\Delta\boldsymbol{p}^h_{j,i}$, and $\bar{\eta}_j$, instead, play the role of buffer (or global) variables that carry the information on the position and orientation of the other agents j. In particular, these buffer variables allow the independent agents to agree on a common strategy to avoid collisions (i.e., to prevent the violation of the collision avoidance constraints (11g)-(11h)). Furthermore, rewrite in a more compact form the equality constraints (12d)-(12e) as follows:

$$A_i\boldsymbol{\xi}_i(t+k) + B_i\boldsymbol{y}_i(t+k) = \boldsymbol{b}_i(t+k), k = 1, \dots, N_i \tag{13}$$

where $\boldsymbol{y}_i := \left[\Delta\boldsymbol{p}_{i,j}^{1^{\mathrm{T}}} \dots \Delta\boldsymbol{p}_{i,j}^{h^{\mathrm{T}}} \Delta\boldsymbol{p}_{j,i}^{1^{\mathrm{T}}} \dots \Delta\boldsymbol{p}_{j,i}^{h^{\mathrm{T}}}\right]^{\mathrm{T}}$. The matrices A_i, B_i, and the vector $\boldsymbol{b}_i$ are defined as follows:

$$A_i := \left[E_i^T \dots E_h^{\mathrm{T}}\, F \dots F\right]^{\mathrm{T}}, \tag{14a}$$

$$B_i := \left[-I \dots -I\, I \dots I\right]^{\mathrm{T}}, \tag{14b}$$

$$\boldsymbol{b}_i^{\mathrm{T}} := \left[\bar{\boldsymbol{p}}_j^{\mathrm{T}} \dots \bar{\boldsymbol{p}}_j^{\mathrm{T}} \bar{\boldsymbol{p}}_j^{1^{\mathrm{T}}} \dots \bar{\boldsymbol{p}}_j^{h^{\mathrm{T}}}\right], \tag{14c}$$

where E^h select $\boldsymbol{p}^h_i$ from the vector of local variables $\boldsymbol{\xi}_i$, and F selects the components of $\boldsymbol{p}_i$ from $\boldsymbol{\xi}_i$. We can rewrite Problem (12) as follows

$$\min_{\boldsymbol{\xi}_i, \boldsymbol{y}_i} \sum_{k=1}^{N} J(\boldsymbol{\xi}_i(t+k)) \tag{15a}$$

$$\text{s.t.: } \boldsymbol{\xi} \in \mathcal{F} \tag{15b}$$

$$A_i\boldsymbol{\xi}_i(t+k) + B_i\boldsymbol{y}_i(t+k) = \boldsymbol{b}_i(t+k), \tag{15c}$$

where $\mathcal{F} := \{\boldsymbol{z} |$ (8b)-(8e), (11e), and (12f)-(12g) are satisfied$\}$ is the feasible region of the local agent, which includes dynamic constraints, control constraints, state constraints, road boundaries, and the reformulation of the local collision avoidance constraints.

Remark 1. Note that the introduction of the nonlinear local equality constraints (11e) (together with a new set of local variables that also impact the local cost with an associated indicator function to ensure the feasibility of the equality constraints) might seem redundant. Their introduction, however, is fundamental for ADMM, which requires linear coupling constraints. In particular, without them constraints (15c) would be nonlinear in the decision variables and there are no existing ADMM algorithms that can handle them to the best of the authors' knowledge.

Algorithm 1 shows the proposed control strategy that relies on ADMM (steps 3-16). In particular, we rely on the modified version of ADMM suitable for nonconvex optimization proposed in [18] due to the presence of the nonconvex constraints (the collision avoidance constraints described in Section II-B) and the nonlinear dynamics of the agents (1), which make the problem to be solved nonconvex.

Compared to [18], the ADMM strategy in Algorithm 1 allows the agents to perform parallel local updates (i.e., the other agents do not have to wait for updated values of the local variables of the neighbors). Neighboring agents, however, still have to communicate to exchange the locally computed values[3] of $\Delta\boldsymbol{p}^h_{i,j}$ (steps 6 and 10). Furthermore, compared to the strategy proposed in [18], we have to deal with constraints in the inner problems solved by the local agents (step 5). Each agent needs a local optimizer able to solve nonlinear nonconvex constrained problems (such as FORCES Pro [15]). In this respect, the augmented Lagrangian associated to Problem (15) in the inner problems is defined as follows:

$$\mathcal{L}(\boldsymbol{\xi}, \boldsymbol{y}, \boldsymbol{\lambda}) := J(\boldsymbol{\xi}) + \langle \boldsymbol{\lambda}, A\boldsymbol{\xi} + B\boldsymbol{y} - b \rangle + \frac{\rho}{2}\|A\boldsymbol{\xi} + B\boldsymbol{y} - b\|^2.$$

The update of the buffer variables is performed in step 9 of Algorithm 1 and it is defined as follows:

$$\Delta\boldsymbol{p}^h_{i,j} := \frac{\boldsymbol{p}^h_i - \bar{\boldsymbol{p}}_j + \bar{\boldsymbol{p}}^h_i - \boldsymbol{p}_j}{2}, \bar{\eta}_j \quad := \eta_j. \tag{16}$$

Note that to perform the update above, it is sufficient for Agent j to communicate the difference $\bar{\boldsymbol{p}}^h_i - \boldsymbol{p}_j$ and η_j along the prediction horizon for all $h \in \mathcal{I}^{\text{disc}}$.

Remark 2. Note that the vector $\boldsymbol{b}_i$ varies along the prediction horizon, but it is not a decision variable. We can precompute its values as follows. Agent i receives/updates iteratively the values of $\Delta\boldsymbol{p}^h_{ij}$, $\Delta\boldsymbol{p}^h_{j,i}$, η_i, and $\boldsymbol{p}_i$. Hence, b_i can be derived from (13) based on the values of $\boldsymbol{z}_i(t+k)$ computed at the previous problem instant, but using the updated values of $\Delta\boldsymbol{p}^h_{ij}, \Delta\boldsymbol{p}^h_{ij}$. We could proceed differently using the values of $\boldsymbol{p}_j$ and $\boldsymbol{p}^h_j$ computed by the neighboring agents. This will lead to an ADMM strategy with more than two sets of variables to update and requires each agent to wait for all the neighboring agents to update their decision variables in a sequential fashion. Our strategy allows all the agents to proceed in parallel with their local computations, saving waiting time and reducing the amount of information to

[3]The values of $\Delta\boldsymbol{p}^h_{i,j}$ are shared among the agents, but for the computation we can use one of the computation units onboard of each agent.

Algorithm 1 Distributed NMPC.

1: Given $\boldsymbol{\xi}_1^0, \ldots, \boldsymbol{\xi}_V^0$, $\boldsymbol{y}_1^0, \ldots, \boldsymbol{y}_V^0$, $\boldsymbol{\lambda}_1^0, \ldots, \boldsymbol{\lambda}_V^0$.
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: **for** $\texttt{iter} = 1, \ldots, \texttt{iter}_{\max}$ **do**
4: **for** $i = 1, \ldots, V$ each agent computes in parallel **do**
5: Update b_i.
6: $\boldsymbol{\xi}_i^{t+1} \leftarrow \operatorname{argmin}_{\boldsymbol{\xi}_i \in \mathcal{F}_i} \mathcal{L}_i\left(\boldsymbol{\xi}_i, \boldsymbol{y}_i^t, \boldsymbol{\lambda}_i^t\right)$.
7: Agent i sends/receives updates from neighbors.
8: **end for**
9: **for** $i = 1, \ldots, V$ each agent computes in parallel **do**
10: $\boldsymbol{y}_i^{t+1} \leftarrow$ according to (16).
11: Agent i sends/receives updates from neighbors.
12: **end for**
13: **for** $i = 1, \ldots, V$ each agent computes in parallel **do**
14: $\boldsymbol{\lambda}_i^{t+1} \leftarrow \boldsymbol{\lambda}_i^t + \rho(A_i \boldsymbol{\xi}_i^{t+1} + B_i \boldsymbol{y}_i^{t+1} - b_i)$.
15: **end for**
16: **end for**
17: **for** $i = 1, \ldots, V$ each agent computes in parallel **do**
18: Select $\boldsymbol{u}_i(1)$ and implement it in closed loop.
19: Update $\boldsymbol{\xi}_i^0$ and return to step 3.
20: **end for**
21: **end for**

share. Furthermore, the direct use of $\boldsymbol{p}_j$ and $\boldsymbol{p}_j^h$ means that the ADMM strategy operates directly on the collision avoidance constraints (that can be converted to equality constraints using nonconvex indicator functions in the cost). This leads to nonlinear equality constraints to be handled by the ADMM solver (with similar consequences to the one pointed out in Remark 1).

V. NUMERICAL RESULTS

For each agent, we consider the vessel model provided in [22], [23] and summarized below:

$$\dot{x} = v_x \cos(\eta) - v_y \sin(\eta) \tag{17a}$$

$$\dot{y} = v_x \sin(\eta) + v_y \cos(\eta) \tag{17b}$$

$$\dot{\eta} = \omega \tag{17c}$$

$$\dot{v}_x = \frac{1}{m}(u_l + u_r - D_x v_x) + \omega v_y \tag{17d}$$

$$\dot{v}_y = -\frac{1}{m} D_y v_y - \omega v_x \tag{17e}$$

$$\dot{\omega} = \frac{1}{I_z}(l(u_l - u_r) - D_\eta \omega) \tag{17f}$$

where the states are the following: *(i)* $\boldsymbol{p} := [x\, y]^{\mathrm{T}}$, that is, the position of the vessel, *(ii)* η, that is, the orientation, *(iii)* v_x and v_y, that are, the velocity in the longitudinal and lateral directions, respectively, and *(iv)* ω, that is, the angular velocity. The commands are the left and right torques, that are, u_l and u_r, respectively. The dynamics of the actuators are not considered at this stage. The commands and the states are subject to the following constraints: $u_l, u_r \in [-686\,\text{kN}, 686\,\text{kN}]$, $v_x \in [0\ \text{m/s}, 1.67\ \text{m/s}]$, $v_y \in [-0.84\ \text{m/s}, 0.84\ \text{m/s}]$, and $\omega \in [-15\pi/180, 15\pi/180]$. Furthermore, we take into account that the states have to stay within the canals' bounds. In this respect, the local constraints of each vessel takes into account also the description of the canal, which is similar to the one described in [8] for the road. Finally, $l = 1$ m, $m = 200$ kg, $I_z = 14$ kg·m/s, $D_x = 38$ kg/s, $D_y = 5280$ kg/s, and $D_\eta = 104$ kg·m^2/s are the vessel's length, mass, moment of inertia along the z axis, and damping parameters, respectively [23]. We consider three vessels (indicated in blue, black, and red in Figure 2) navigating at a canal intersection. In particular, we consider a scenario in which the blue vessel has to turn left, the black vessel has to turn left, and the red vessel has to go straight ahead.

The challenge is for the vessels to cross the intersection safely, that is, without colliding with the other two vessels. The vessels have to agree on which vessel should cross first between the blue and the black one, and which vessel should go ahead first between the blue and the red one.

To reduce the amount of information exchange among the vessel, we limited the number of exchanges within the sampling time of the system to $\texttt{iter}_{\max} = 4$. In order to improve the performance of the algorithm (given the limited exchange of information), at every sampling instance, the algorithm is warm-started with the results available at the previous sampling instance. Furthermore, we fix the value of ρ to 0.005 and we select a prediction horizon of $N_i = 50$. The inner problems (step 5 of Algorithm 1) are solved using FORCES Pro [15] in Matlab R2016b running on a Windows OS with an Intel (R) Xeon (R) CPU @3.40 GHz.

Figure 2 summarizes the results obtained using the proposed control strategy. In particular, row (1) shows the trajectories of the vessels while crossing the intersection. Row (2) shows the longitudinal velocity of the vessels. Finally, row (3) shows the control command.

As Figure 2 shows, during phase (a) the vessels approach the intersection. Then, during phase (b), as depicted row (2) of Figure 2, the black vessel reduces its speed and allows the blue vessel to occupy intersection. During phase (c), the blue vessel reduces its speed to turn and interacts with the red vessel to agree on the boat that will take the lead. Finally, during phase (d), the blue vessel increases its speed and proceeds in front of the red vessel.

From the computation point of view, the current MATLAB implementation requires, for each vessel to agree on a control strategy (steps 3-16 of Algorithm 1, on average less than 0.8 sec and has a worst-case computation time of 1.2 sec.

Thanks to the proposed problem decomposition, the vessels are able to make decisions autonomously without the need for a central coordinator.

VI. CONCLUSIONS AND FUTURE WORK

We proposed a distributed model predictive control algorithm for the coordination of autonomous agents. By relying on a state-of-the-art alternating direction method of multipliers suitable for nonconvex optimization, the proposed algorithm allows the agents to communicate (sharing a limited amount of information) and agree on a common safe (i.e., collision-free) navigation strategy without the need of a central coordinator. We tested the proposed design for the control of vessels at a canal intersection. Nevertheless, the proposed design is general and can be applied for the distributed coordination of cars, aircraft, etc.

The current implementation considers that at every sampling time, the vessels will stop to run steps 3-16 of Algo-

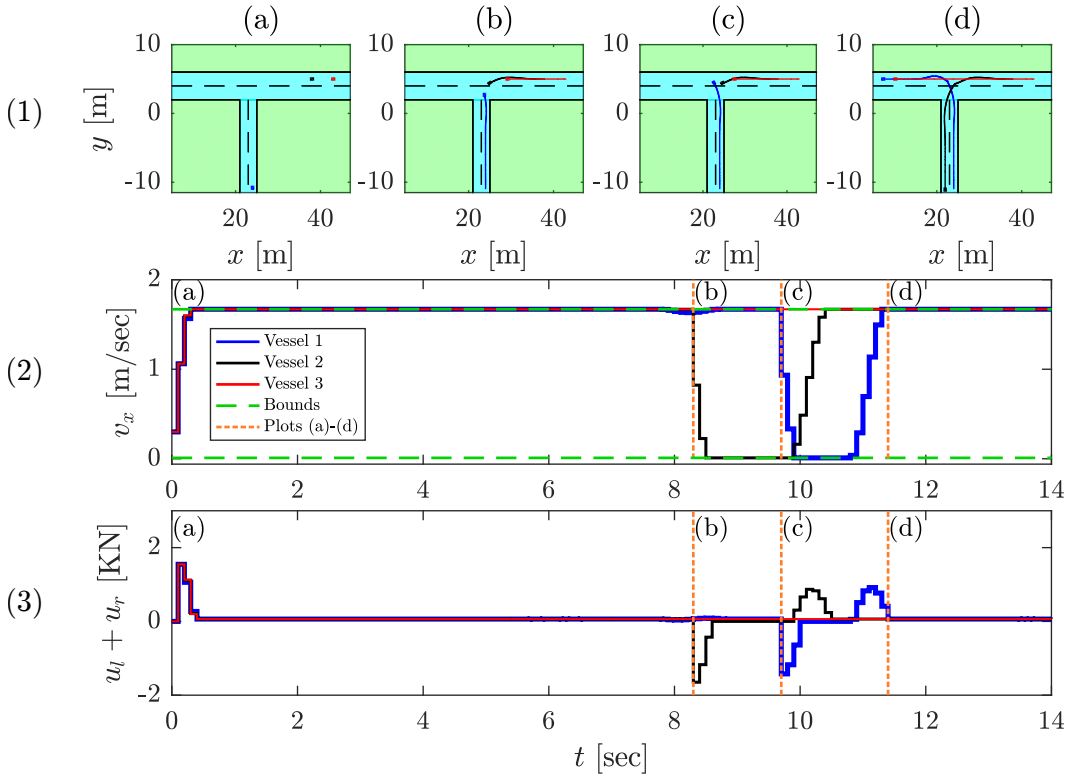


Fig. 2: Behavior of three vessels crossing an intersection using Algorithm 1.

rithm 1 and will proceed only when the algorithm terminates. As part of our future work, we plan to investigate more practical strategies (by relying on asynchronous communications) to run Algorithm 1 in real-time, that is, without the vessel stopping to compute the control law.

REFERENCES

[1] M. R. Hafner, D. Cunningham, L. Caminiti, and D. D. Vecchio, "Cooperative collision avoidance at intersections: Algorithms and experiments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1162–1175, 2013.

[2] H. Ahn and D. Del Vecchio, "Safety verification and control for collision avoidance at road intersections," *arXiv preprint arXiv:1612.02795*, 2016.

[3] G. R. de Campos, P. Falcone, R. Hult, H. Wymeersch, and J. Sjöberg, "Traffic coordination at road intersections: Autonomous decision-making algorithms using model-based heuristics," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 8–21, 2017.

[4] A. Colombo and D. D. Vecchio, "Least restrictive supervisors for intersection collision avoidance: A scheduling approach," *IEEE Transactions on Automatic Control*, vol. 60, no. 6, pp. 1515–1527, 2015.

[5] M. Zanon, S. Gros, H. Wymeersch, and P. Falcone, "An asynchronous algorithm for optimal vehicle coordination at traffic intersections," in *20th IFAC World Congress*, 2017.

[6] J. Bento, N. Derbinsky, J. Alonso-Mora, and J. S. Yedidia, "A message-passing algorithm for multi-agent trajectory planning," in *Proceedings of NIPS*, 2013, pp. 521–529.

[7] D. Hrovat, S. Di Cairano, H. E. Tseng, and I. V. Kolmanovsky, "The development of model predictive control in automotive industry: A survey," in *Proceedings of the IEEE International Conference on Control Applications*, Oct. 2012, pp. 295–302.

[8] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention," in *Proceedings of the IEEE ICRA*, 2017.

[9] H. Zheng, R. R. Negenborn, and G. Lodewijks, "Cooperative distributed collision avoidance based on ADMM for waterborne AGVs," in *Proceedings of the ICCL'15*, 2015, pp. 181–194.

[10] ——, "Fast ADMM for distributed model predictive control of cooperative waterborne AGVs," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1406–1413, 2017.

[11] L. Ferranti and T. Keviczky, "Operator-splitting and gradient methods for real-time predictive flight control design," *Journal of Guidance, Control, and Dynamics*, pp. 1–13, 2016, DOI:10.2514/1.G000288.

[12] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *Journal of Optimization Theory and Applications*, vol. 99, no. 3, pp. 723–757, 1998.

[13] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009, DOI:10.1137/080716542.

[14] Y. Nesterov, "Smooth minimization of non-smooth functions," *Mathematical Programming*, vol. 103, no. 1, pp. 127–152, 2005.

[15] A. Domahidi and J. Jerez, "FORCES Professional," embotech GmbH (http://embotech.com/FORCES-Pro), Jul. 2014.

[16] T. Faulwasser, B. Kern, and R. Findeisen, "Model predictive path-following for constrained nonlinear systems," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, 2009, pp. 8642–8647.

[17] D. Lam, C. Manzie, and M. C. Good, "Model predictive contouring control for biaxial systems," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 2, pp. 552–559, 2013.

[18] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *arXiv preprint arXiv:1511.06324*.

[19] H. Zheng, "Coordination of waterborne AGVs," Ph.D. dissertation, Delft University of Technology, 2016, TRAIL Research School.

[20] J. Xin, "Control and coordination for automated container terminals," Ph.D. dissertation, TU Delft, 2015, TRAIL Research School.

[21] 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.

[22] T. I. Fossen, *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[23] G. Hitz, F. Pomerleau, F. Colas, and R. Siegwart, "Relaxing the planar assumption: 3D state estimation for an autonomous surface vessel," *The International Journal of Robotics Research*, vol. 34, no. 13, pp. 1604–1621, 2015.