Asynchronous Distributed Collision Avoidance With Intention Consensus for Inland Autonomous Ships

Hoang Anh Tran, Nikolai Lauvås, Tor Arne Johansen, *Senior Member, IEEE*, and Rudy R. Negenborn

***Abstract*—This article focuses on the problem of collaborative collision avoidance (CCAS) for autonomous inland ships. Two solutions are provided to solve the problem in a distributed manner. We first present a distributed model predictive control (MPC) algorithm that allows ships to directly negotiate their intention to avoid collision in a synchronous communication framework. Moreover, we introduce a new approach to shape the ship's behavior to follow the waterway traffic regulations. The conditional convergence toward a stationary solution of this algorithm is guaranteed by the theory of the alternating direction method of multipliers (ADMM). To overcome the problem of asynchronous communication between ships, we adopt a new asynchronous nonlinear ADMM (Async-NADMM) and present an asynchronous distributed MPC algorithm based on it. Several simulations and field experiments show that the proposed algorithms can guarantee a safe distance between ships in complex scenarios while following the traffic regulations. Furthermore, the asynchronous algorithm has an efficient computational time and satisfies the real-time computing requirements of ships in field experiments.**

***Index Terms*—Autonomous ship, collision avoidance, distributed control, model predictive control (MPC), optimal control, ship and vessel control.**

I. Introduction

THE recent decade has witnessed an increase in research toward inland autonomous ships. One of the most critical components of an autonomous ship is the collision avoidance system (CAS), which ensures the ship's collision-free navigation. Several approaches have been proposed to increase the navigational safety of ships, such as model predictive control (MPC) algorithms [1], [2], scenario-based MPC [3], [4], and velocity obstacles [5]. That being said, many of these studies are reactive methods that do not explicitly consider changes in neighboring ships' intentions.

Received 1 May 2025; accepted 4 July 2025. Date of publication 23 July 2025; date of current version 23 October 2025. The work of Hoang Anh Tran and Tor Arne Johansen was supported by European Union's Horizon 2020 Research and Innovation Program through the Marie Skłodowska-Curie under Grant 955.768 (MSCA-ETN AUTOBarge). The work of Nikolai Lauvås was supported by the NTNU Fish Otter Project. The work of Rudy R. Negenborn was supported by the Researchlab Autonomous Shipping at Delft University of Technology. Recommended by Associate Editor C. Vermillion. *(Corresponding author: Hoang Anh Tran.)*

Hoang Anh Tran, Nikolai Lauvås, and Tor Arne Johansen are with the Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway (e-mail: hoang.a.tran@ntnu.no; nikolai.lauvas@ntnu.no; tor.arne.johansen@ntnu.no).

Rudy R. Negenborn is with the Department of Maritime and Transport Technology, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: r.r.negenborn@tudelft.nl).

Digital Object Identifier 10.1109/TCST.2025.3587842

There are two common approaches to dealing with the change in intentions of the neighboring ships. One approach is to predict the intentions of neighboring ships [6]. This approach uses a dynamic Bayesian network to model and predict the intentions of neighboring ships based on the observation of real-time behavior of neighboring ships. However, these algorithms cannot always guarantee high accuracy in the prediction. The other approach is to exchange intentions between ships through wireless communication. This approach removes the ambiguity of neighboring ships' intentions but requires ships to be aided with sufficient communication equipment [7]. However, the rapid development of communication technology and protocols makes sharing of intentions between ships practical [8], [9].

A framework is needed for ships to collaborate efficiently when exchanging intentions. Different solutions have been proposed for a framework of collaborative collision avoidance (CCAS) of ships [10], [11], [12]. The CCAS framework can be categorized as centralized or distributed based on the architecture of the communication network. The difference between the two frameworks is the actor that executes the CCAS. In the centralized framework, the CCAS algorithm is performed by one specific coordinator, e.g., the traffic control center or one ship with powerful computation equipment, and the solution is sent to each ship [13], [14]. The ships are assumed to follow up precisely. This framework requires fewer onboard computational resources, but ships are vulnerable to communication faults, i.e., loss of communication with the coordinator. On the contrary, the distributed framework requires all ships to perform the CCAS algorithm individually and negotiate with each other to reach a consensus solution [15], [16]. In case of communication loss between ships, the CCAS algorithm onboard can act as a conventional CAS without intention sharing. Although the distributed framework requires more onboard computational resources, the robustness under unstable communication makes this framework more promising for future applications in CCAS [7].

In CAS for autonomous vehicles, the MPC algorithm is widely used due to its ability to foresee and resolve the potential risks in complex traffic scenarios [17], [18], [19]. A common approach to implement MPC algorithms in the distributed CCAS in autonomous ships is using the alternating direction method of multipliers (ADMM) [11], [13], [20]. The ADMM method was initially developed for the distributed optimization of a convex problem [21]. The ADMM algorithm combines the advantages of dual decomposition

and augmented Lagrangian methods to address constrained convex optimization problems. The ADMM and its variants are widely used in machine learning and statistical modeling [21]. However, the assumption of convexity may restrict the applications of the ADMM in CCAS. Themelis and Patrinos [22] presented a universal equivalent between the nonconvex Douglas–Rachford splitting method and ADMM for a class of nonconvex optimization problems. This finding not only allows for the direct application of nonlinear ADMM (NADMM) in CCAS [19], [23] but also opens an opportunity to apply the theory of Douglas–Rachford splitting to NADMM algorithms.

When using ADMM to solve the distributed CCAS problem, one important factor is whether the communication is synchronous or asynchronous. In a synchronous communication network, controllers must wait for each other to acknowledge the new update before continuing. On the other hand, a controller in an asynchronous communication network can send updates continuously without waiting for others. Although the distributed ADMM attains linear and global convergence in synchronous networks [24], [25], the synchronous communication in CCAS for autonomous ships is either hard to implement or can delay the process. Several approaches have been proposed to tackle the problem of asynchronous convex ADMM [26], [27]. However, there are limited results for the nonconvex ADMM problems. In [28] and [29], two asynchronous ADMM approaches for nonconvex problems for a specific type of equality constraint are presented. In this article, equality constraints represent the consensus of decisions between ships, i.e., future trajectories. They are not of the same type as those presented in [28] and [29]. Therefore, we propose an asynchronous NADMM (Async-NADMM) algorithm that uses a more general equality constraint and extends the aforementioned algorithms.

While several studies have addressed the problem of CCAS for autonomous vehicles, there are limited results in the field of autonomous inland ships. To the best of our knowledge, no proposed algorithm has defined an asynchronous CCAS framework for ships that considers inland waterway traffic regulations. This article presents two approaches to solving the problem of distributed CCAS for autonomous ships in inland waterways. The main contributions of this article are as follows.

1) We present the synchronous CCAS (Sync-CCAS), a distributed MPC for the CCAS problem considering synchronous communication between ships. In addition, this algorithm explicitly considers inland waterway traffic regulations.
2) To deal with the problem of asynchronous communication in practical conditions, we present the Async-NADMM algorithm. This algorithm is based on the asynchronous randomized Douglas–Rachford splitting (asynFedDR) algorithm. Furthermore, we extend the condition for the equality constraints to fit our CCAS problem.
3) We introduce the asynchronous CCAS (Async-CCAS), which has the same properties as the Sync-CCAS algorithm, but concerns the case of asynchronous communication, with signal delays between ships.

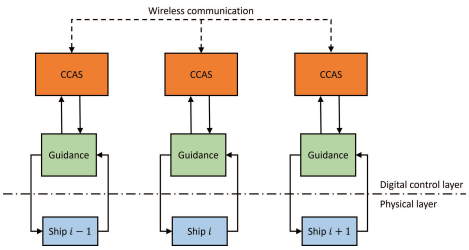


Fig. 1. Collaborative collision avoidance scheme between ships.

We consider the set $\mathcal{M}$ of M ships to solve the CAS problem collaboratively in a distributed manner. The consensus of intentions between ships is reached through negotiation using digital ship-to-ship communication.

Fig. 1 shows the control scheme for autonomous ship i with the CCAS in the loop. The guidance system keeps the ship on track with the predefined waypoints by commanding the control system to maintain the desired speed and course angle. The CCAS continuously checks for potential collision risks on the sailing path. If a risk of collision is detected, the CCAS will perform the necessary course change and/or speed change modification to avoid the collision and achieve compliance with traffic rules when possible. Besides, the CCAS also exchanges the predicted trajectories with neighboring ships. The CCAS includes the following three main components.

1) *Kinematic Model of Ships:* To predict the future position of the own and the neighboring ships.
2) *Collision Risk Evaluation:* To evaluate the situation and detect potential collision risks and violations of traffic rules based on predicted future trajectories of all ships.
3) *MPC-Based Collision Avoidance Algorithm:* To determine the necessary evasive maneuver based on the potential risks. Since the CCAS contains only one algorithm, the MPC-based collision avoidance algorithm will be called the CCAS algorithm.

We adopt the NADMM in [22] to develop the synchronous CCAS algorithm, where its preliminary version was presented in [15]. Compared to [15], we introduce a new approach to optimize ship behavior following waterway traffic regulations. The new approach is better at handling cases where neither ship has a duty to give way and needs to negotiate to avoid collision. Moreover, this article provides a theoretical analysis of the convergence of the synchronous algorithm. To deal with asynchronous communication, we present the Async-NADMM algorithm for nonconvex optimization problems with a relaxed equality constraint. Due to the equivalence between the nonconvex Douglas–Rachford splitting method and NADMM, we can cast the result of the asynchronous Douglas–Rachford splitting method in [30] to the NADMM problem. Then, the Async-CCAS is formed based on the Async-NADMM. The performance of the proposed algorithms is verified in both simulation and field experiments with various representative traffic scenarios.

The remainder of this article is structured as follows. Section II presents preliminary information needed to

formulate the CCAS in the later section, including notations, definitions, and the components of the CCAS. The Sync-CCAS and its convergence analysis are presented in Section III. Section IV introduces the Async-ADMM and, with it, the Async-CCAS. Finally, the experimental results are presented in Section V, and Section VII concludes this article.

II. Preliminaries

This section presents notations and definitions that will be used. We also briefly present the first two components of the CCAS that provide the background for the proposed CCAS algorithms. Additionally, this section presents the NADMM, which is used as a framework for the Sync-CCAS.

A. Notations and Definitions

We denote **range**(M) as the range (column space) of the matrix M. The domain of an extended-real-valued function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is **dom**$(f) := \{x \in \mathbb{R}^n \mid f(x) < \infty\}$.

Definition 1 (Lower Semicontinuous Function): A function $f : X \to \bar{\mathbb{R}}$ is called lower semicontinuous (lsc) at point $x_0 \in X$ if $\liminf_{x \to x_0} f(x) = f(x_0)$. Furthermore, f is called lsc if $f(x)$ is lsc at every point $x_0 \in$ **dom**(f).

Definition 2 (Lipchitz Continuous Gradient): A differentiable function h is said to have Lipschitz continuous gradient with constant $L_h > 0$ (or L_h – smooth) on **dom**(h) if

$$\|\nabla h(x_1) - \nabla h(x_2)\| \le L_h \|x_1 - x_2\| \quad \forall x_1, x_2 \in \mathbf{dom}(h).$$

Definition 3 (Stationary Point): Given $f : \mathbb{R}^n \to \bar{\mathbb{R}}$, then $x^* \in$ **dom**(f) is called an stationary point of $f(x)$ if $\nabla f(x^*) = 0$.

Definition 4 (Image Function): Given $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ and $M \in \mathbb{R}^{m \times n}$, then the image function $(Mf) : \mathbb{R}^m \to [-\infty, +\infty]$ is defined as $(Mf)(\epsilon) := \inf_{x \in \mathbb{R}^n}\{f(x) \mid Mx = \epsilon\}$.

Definition 5 (Proximal Mapping): Given $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ and $\gamma > 0$, then the proximal mapping $\mathbf{prox}_{\gamma f} : \mathbb{R}^n \to$ **dom**(f) is defined as

$$\mathbf{prox}_{\gamma f}(x) = \arg\min_y \left\{ f(y) + \frac{1}{2\gamma}\|y - x\|^2 \right\}. \tag{1}$$

If f is L_f-smooth and $\gamma < L_f$, then $\mathbf{prox}_{\gamma f}(x)$ is well-defined and single valued. Furthermore, as pointed out in [22], we also have the following property:

$$\frac{y - x}{\gamma} \in \nabla f(x) \quad \forall x = \mathbf{prox}_{\gamma f}(y). \tag{2}$$

In this article, the path coordinate frame ϖ_i is used by ship $i \in \mathcal{M}$ to represent the state of itself and other ships in the network $\mathcal{M}$. We denote $p_{j,i} = [x_{j,i}, y_{j,i}, \chi_{j,i}]^\top$ as the state of ship j with respect to coordinate frame $\{\varpi_i\}$, consisting of coordinates along and across guideline between waypoints, and the angle relative to guideline. The relation between $p_{j,i}$ and the state of ship j in the inertial coordinate frame $\{n\}$, i.e., $\eta_j = [x_{j,n}, y_{j,n}, \chi_{j,n}]^\top$, is as follows:

$$p_{j,i} = \begin{bmatrix} x_{j,i} \\ y_{j,i} \\ \chi_{j,i} \end{bmatrix} = R_i \begin{bmatrix} x_{j,n} - x_{i,n}^{\mathrm{WP}} \\ y_{j,n} - y_{i,n}^{\mathrm{WP}} \\ \chi_{j,n} - \chi_{i,n}^{\mathrm{WP}} \end{bmatrix} = R_i \left(\eta_j - \eta_i^{\mathrm{WP}}\right)$$

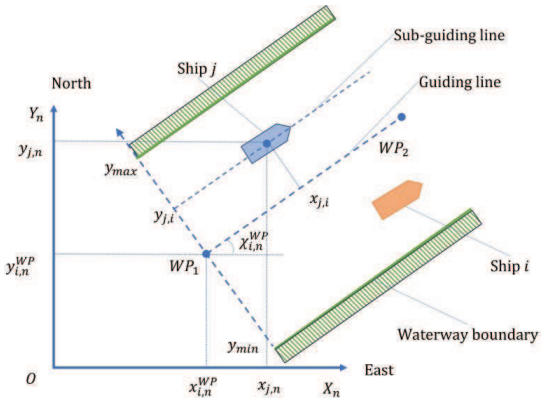


Fig. 2. Path coordinate frame $\{\varpi_i\}$, with origin at WP1 and inertial coordinate frame $\{n\}$.

$$R_i = \begin{bmatrix} \cos\left(\chi_{i,n}^{\mathrm{WP}}\right) & \sin\left(\chi_{i,n}^{\mathrm{WP}}\right) & 0 \\ -\sin\left(\chi_{i,n}^{\mathrm{WP}}\right) & \cos\left(\chi_{i,n}^{\mathrm{WP}}\right) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where $\eta_i^{\mathrm{WP}} = [x_{i,n}^{\mathrm{WP}}, y_{i,n}^{\mathrm{WP}}, \chi_{i,n}^{\mathrm{WP}}]^\top$ is the parameter of the previous active waypoint in the inertial frame (see Fig. 2).

B. Kinematic Model of Ships

1) Kinematic Model of Own-Ship: In this article, the kinematic model of the ship is the combination of the ship's physical characteristics with the guidance system and the low-level controllers. The low-level controllers, the speed and heading controllers, are assumed to have a fast time constant to attain the control command from the CCAS.

The kinematic model of ship i with respect to coordinate frame $\{\varpi_i\}$ is described according to [31] as follows:

$$\begin{aligned} x_{i,i}(k+1) &= x_{i,i}(k) + u_{i,i}^s(k) U_i^d \cos(\chi_{i,i}(k))\Delta T \\ y_{i,i}(k+1) &= y_{i,i}(k) + u_{i,i}^s(k) U_i^d \sin(\chi_{i,i}(k))\Delta T \\ \chi_{i,i}(k+1) &= \chi_{i,i}(k) + \frac{\Delta T}{T_1} \\ &\quad \cdot \left[\chi_i^{\max} \tanh(K_e(u_{i,i}^y(k) - y_{i,i}(k))) - \chi_{i,i}(k)\right] \end{aligned} \tag{3}$$

where U_i^d is the nominal surge speed; $\chi_i^{\max}$ is the maximum course angle change that ship i can achieve within a sampling period ΔT; and K_e and T_1 are the tuning parameters of the guidance law, in this case, the line-of-sight guidance law. The control inputs of model (3) are the cross-track offset $u_{i,i}^y$ and speed modification $u_{i,i}^s$. Hence, $u_{i,i}(k) = [u_{i,i}^y(k), u_{i,i}^s(k))]^\top$ denotes the vector of the control signal of ship i.

2) Kinematic Model of Neighboring Ships: The kinematic model of ship j with respect to the coordinate frame $\{\varpi_i\}$ can be written as follows:

$$\begin{aligned} x_{j,i}(k+1) &= x_{j,i}(k) + u_{j,i}^s(k) U_j(k) \cos(\chi_{j,i}(k))\Delta T \\ y_{j,i}(k+1) &= y_{j,i}(k) + u_{j,i}^s(k) U_j(k) \sin(\chi_{j,i}(k))\Delta T \\ \chi_{j,i}(k+1) &= \chi_{j,i}(k) + \frac{\Delta T}{T_j}\left(\chi_{j,i}^d(k) + \chi_{j,i}^{\mathrm{nom}} - \chi_{j,i}(k)\right) \end{aligned} \tag{4}$$

where U_j is the nominal surge speed of ship j and T_j is the time constant of the course dynamics. Additionally, $\chi_{j,i}^{\mathrm{nom}} = \chi_{j,n}^{\mathrm{WP}} - \chi_{i,n}^{\mathrm{WP}}$ is the angle of the active reference course of ship

j, i.e., $\chi^{\mathrm{WP}}_{j,n}$, with respect to coordinate frame $\{\varpi_i\}$. The vector of input signals is $u_{j,i}(k) = [u^s_{j,i}(k), \chi^d_{j,i}(k)]^\top$, where $u^s_{j,i}(k)$ and $\chi^d_{j,i}(k)$ are the speed and course adjustments, respectively. In this article, we assume that ship i gets the information of the model (4) of ship j through communication.

For sake of notational simplicity, let us rewrite models (3) and (4), respectively, as follows:

$$p_{i,i}(k+1) = f_{i,i}(p_{i,i}(k), u_{i,i}(k)) \tag{5}$$

$$p_{j,i}(k+1) = f_{j,i}(p_{j,i}(k), u_{j,i}(k)). \tag{6}$$

C. Collision Risk Evaluation

The prediction of the collision risk of ship i concerning ship j at k time steps from the present time t_0 is $R_{ij}(t_0+k)$. According to [23], the collision risk function $R_{ij}(t_0+k)$ is as follows:

$$\begin{aligned} R_{ij}(t_0+k) &= \frac{K_{\mathrm{ca}}}{\sqrt{1+K_d k}} D_x(t_0+k) D_y(t_0+k) \\ D_x(t_0+k) &= \exp\left[-\frac{\left(x_{i,i}(t_0+k) - x_{j,i}(t_0+k)\right)^2}{\alpha_{xj}}\right] \\ D_y(t_0+k) &= \exp\left[-\frac{\left(y_{i,i}(t_0+k) - y_{j,i}(t_0+k)\right)^2}{\alpha_{yj}}\right] \end{aligned} \tag{7}$$

where K_{ca} is a predefined constant determined by safety criteria, which varies according to the traffic situation; $(1/(1+K_d k)^{1/2})$, with $K_d \geq 0$ being a discount factor that reduces the collision risks associated with larger k. Additionally, α_{xj} and α_{yj} are parameters characterizing the size and shape of ship j. In practice, α_{xj} and α_{yj} are chosen by trial and error until the ship reaches an expected behavior.

From the definition of $R_{ij}(t_0+k)$, its value is approximately 0 if the distance between ship i and j is large enough, and increases if this distance is reduced. By minimizing the value of $R_{ij}(t_0+k)$, we can increase the navigation safety of ships.

D. Nonlinear ADMM

Given the optimization problem as follows:

$$\min_{(w,v)\in\mathbb{R}^m\times\mathbb{R}^n} f(w) + g(v) \tag{8a}$$

$$\text{s.t. } Aw - Bv - c = 0 \tag{8b}$$

where $f\colon \mathbb{R}^m \to \bar{\mathbb{R}}$, $g\colon \mathbb{R}^n \to \bar{\mathbb{R}}$, $A \in \mathbb{R}^{q\times m}$, $B \in \mathbb{R}^{q\times n}$, and $c \in \mathbb{R}^q$. The augmented Lagrangian of problem (8) with the Lagrange multiplier $z \in \mathbb{R}^q$ is

$$\begin{aligned} \mathcal{L}_\beta(w,v,z) = f(w) + g(v) + \langle z, Aw - Bv - c\rangle \\ + \frac{\beta}{2}\|Aw - Bv - c\|^2. \end{aligned}$$

Then, the NADMM algorithm in [22] solves problem (8) iteratively as

$$\begin{cases} z^{+1/2} = z - \beta(1-\lambda)(Aw - Bv - c) \\ w^+ \in \arg\min \mathcal{L}_\beta\left(\cdot, v, z^{+1/2}\right) \\ z^+ = z^{+1/2} + \beta(Aw - Bv - c) \\ v^+ \in \arg\min \mathcal{L}_\beta\left(w, \cdot, z^+\right) \end{cases} \tag{9}$$

where $\beta > 0$ and $\lambda \in (0,2)$ are tuning parameters.

III. Synchronous Distributed Collaborative Collision Avoidance Algorithm

This section presents the Sync-CCAS algorithm. The Sync-CCAS is developed based on the assumption of synchronous communication, that is, after every negotiation round, each ship waits for the decisions of all neighboring ships before starting the next round.

A. Problem Formulation

The problem of CCAS for autonomous ships is formulated as a distributed MPC problem. The cost function combines the potential risk and the cost of taking action. By optimizing the cost function, we aim to find a safe trajectory that complies with traffic rules and with minimum effort.

First, we made the following assumptions.

Assumption 1: For every pair i and j of ships in the set $\mathcal{M}$, there exists a communication link between ship i and j, and the bandwidth of the communication link is sufficient for ships to exchange information. Through communication and sensor information of their own, all ships in the set $\mathcal{M}$ share the same information of the traffic situation.

Assumption 2: All ships in the set $\mathcal{M}$ are aided with capable computational equipment to compute the solution of the optimization problem.

For simplicity, we denote $\tilde{u}_{i,i} = [u^\top_{i,i}(t_0), u^\top_{i,i}(t_0+1), \ldots, u^\top_{i,i}(t_0+N-1)]^\top$ and $\tilde{p}_{i,i} = [p^\top_{i,i}(t_0), p^\top_{i,i}(t_0+1), \ldots, p^\top_{i,i}(t_0+N)]^\top$ as, respectively, control input and state vectors of ship i over a control horizon of N time steps. Moreover, each ship $i \in \mathcal{M}$ stores a local copy of the position of all neighboring ships with respect to the coordinate frame $\{\varpi_i\}$, denoted as $\tilde{p}_i = [\tilde{p}^\top_{1,i}, \tilde{p}^\top_{2,i}, \ldots, \tilde{p}^\top_{M,i}]^\top$. Similarly, $\tilde{u}_i = [\tilde{u}^\top_{1,i}, \tilde{u}^\top_{2,i}, \ldots, \tilde{u}^\top_{i,i}, \ldots, \tilde{u}^\top_{M,i}]^\top$ is ship i's prediction of the control input of all ships in $\mathcal{M}$. Besides the local variable $\tilde{p}_i$, we define ξ as the global variable that stores the consensus trajectory of all ships over N time steps. The relation between the local and the global variable is

$$\tilde{p}_i = \tilde{R}_i\left(\xi - \bar{\eta}^{\mathrm{WP}}_i\right) \quad \forall i \in \mathcal{M} \tag{10}$$

where $\bar{\eta}^{\mathrm{WP}}_i = \mathbf{1}_{M(N+1)} \otimes \eta^{\mathrm{WP}}_i$ with $\otimes$ denoting the Kronecker product and $\tilde{R}_i$ is defined as follows:

$$\tilde{R}_i = \begin{bmatrix} R_i & & 0 \\ & \ddots & \\ 0 & & R_i \end{bmatrix} \in \mathbb{R}^{3(N+1)M\times 3(N+1)M}.$$

Then, we formulate the cost function for the MPC problem of ship i as follows:

$$\mathcal{J}_i(\tilde{p}_i, \tilde{u}_i) = \mathcal{J}^{\mathrm{ca}}_i(\tilde{p}_i) + \mathcal{J}^e_i(\tilde{u}_i) + \mathcal{J}^b_i(\tilde{u}_i). \tag{11}$$

Here, $\mathcal{J}^{\mathrm{ca}}_i(\tilde{p}_i) = \sum_{k=1}^{N+1}\sum_{j\in\mathcal{M}\setminus\{i\}} R_{ij}(t_0+k)$ is the cumulative sum of risk functions with all neighboring ships over the horizon. $\mathcal{J}^e_i(\tilde{u}_i)$ is the cost of control actions, defined as

$$\begin{aligned} \mathcal{J}^e_i(\tilde{u}_i) = \sum_{k=1}^{N+1}\Big[K_y\left(u^y_{i,i}(t_0+k) - u^y_{i,i}(t_0+k-1)\right)^2 \\ + K_s\left(1 - u^s_{i,i}(t_0+k)\right)^2\Big] \end{aligned}$$

$$+\sum_{j\in\mathcal{M}\backslash\{i\}}\sum_{k=1}^{N+1}\alpha_{j,i}\left[\left(\chi_{j,i}^{d}(k)\right)^{2}+\left(1-u_{j,i}^{s}(k)\right)^{2}\right] \quad (12)$$

where K_y and K_s are positive tuning parameters. The first term in (12) is the cost of ship i's action and indicates that ship i should only change its course if it can significantly lower the risk of a collision. The second term in (12) is the cost of proposing course changes to neighboring ships. Depending on the priority constant $\alpha_{j,i} > 0$, ship i can decide to change course or propose ship j to change course. Details on choosing appropriate $\alpha_{j,i}$ are discussed in Section III-B. Finally, $\mathcal{J}_i^b(\tilde{u}_i)$ is a term that makes the behavior of ship i adequately represent waterway traffic regulations, e.g., preferring to steer toward starboard in a head-on situation. More details on $\mathcal{J}_i^b(\tilde{u}_i)$ can be found in [23].

We formulate the distributed MPC collision avoidance problem of ship $i \in \mathcal{M}$, with cost function (11) as follows:

$$\min_{\tilde{p}_i,\tilde{u}_i} \mathcal{J}_i(\tilde{p}_i,\tilde{u}_i) \quad (13a)$$

$$\text{s.t. } p_{i,i}(t_0+k+1) = f_{i,i}\left(p_{i,i}(t_0+k), u_{i,i}(t_0+k)\right) \quad (13b)$$

$$p_{j,i}(t_0+k+1) = f_{j,i}\left(p_{j,i}(t_0+k), u_{j,i}(t_0+k)\right) \quad (13c)$$

$$p_{j,i}(t_0) = p_{j,i}^{\text{init}} \quad (13d)$$

$$u_{i,i}(t_0+k) \in \mathcal{U}_i \quad (13e)$$

$$u_{j,i}(t_0+k) \in \mathcal{U}_{j,i} \quad (13f)$$

$$\tilde{p}_i = \tilde{R}_i\left(\xi - \bar{\eta}_i^{\text{WP}}\right) \quad (13g)$$

$\forall j \in \mathcal{M}$, where $\mathcal{U}_i$ and $\mathcal{U}_{j,i}$ are the admissible sets of $u_{i,i}$ and $u_{j,i}$, respectively. To ensure that ship i cannot propose another ship to steer port side, we bound $\chi_{j,i}^d(k)$ below by 0.

To fit problem (13) into the NADMM framework, we define $\mathbf{G}_i := \{[\tilde{u}_i^\top, \tilde{p}_i^\top]^\top | (13b) -- (13f)\ aresatisfied\}$ as the feasible region for states and inputs of ship i. Then, for ship $i \in \mathcal{M}$, the NADMM update at iteration with index s is defined as follows:

$$\xi^{s+1} = \frac{1}{M}\sum_{j=1}^{M}\hat{\xi}_j^s \quad (14a)$$

$$z_i^{s+1/2} = z_i^s - \beta(1-\lambda)\left(\tilde{R}_i^{-1}\tilde{p}_i^s + \bar{\eta}_i^{\text{WP}} - \xi^{s+1}\right) \quad (14b)$$

$$\begin{bmatrix}\tilde{u}_i^{s+1}\\ \tilde{p}_i^{s+1}\end{bmatrix} = \underset{[\tilde{u}_i^\top,\tilde{p}_i^\top]^\top\in\mathbf{G}_i}{\text{argmin}}\ \{\mathcal{L}_i(\tilde{p}_i,\tilde{u}_i)\} \quad (14c)$$

$$z_i^{s+1} = z_i^{s+1/2} + \beta\left(\tilde{R}_i^{-1}\tilde{p}_i^{s+1} + \bar{\eta}_i^{\text{WP}} - \xi^{s+1}\right) \quad (14d)$$

$$\hat{\xi}_i^{s+1} = \tilde{R}_i^{-1}\tilde{p}_i^{s+1} + \bar{\eta}_i^{\text{WP}} + \frac{1}{\beta}z_i^{s+1} \quad (14e)$$

where

$$\mathcal{L}_i(\tilde{p}_i,\tilde{u}_i) = \mathcal{J}_i(\tilde{p}_i,\tilde{u}_i) + \left\langle z_i^{s+1/2}, \left(\tilde{R}_i^{-1}\tilde{p}_i + \bar{\eta}_i^{\text{WP}} - \xi^{s+1}\right)\right\rangle + \frac{\beta}{2}\left\|\left(\tilde{R}_i^{-1}\tilde{p}_i + \bar{\eta}_i^{\text{WP}} - \xi^{s+1}\right)\right\|^2$$

with z_i being the vector of Lagrange multipliers. The detailed steps of the Sync-CCAS algorithm are presented in Algorithm 1. Here, the local update (14c) is treated as a regular constrained optimization problem with the cost function $\mathcal{L}_i(\tilde{p}_i,\tilde{u}_i)$ and constraint set $\mathbf{G}_i$. It is worth noting that if there is a change in the number of participant ships in Algorithm 1, e.g., a new ship enters the traffic situation, then problem (13) needs to be reformulated.

Algorithm 1 Sync-CCAS

1: **for** $s = 1, ..., s_{\max}$ **do**
2: **for all** $i \in \mathcal{M}$ in parallel **do**
3: Receive $\hat{\xi}_j^s$ from neighboring ships
4: Update the global variable ξ^{s+1} using (14a)
5: Update local variables z_i^{s+1}, $\tilde{u}_i^{s+1}$, $\tilde{p}_i^{s+1}$ and $\hat{\xi}_j^{s+1}$ using (14b)–(14e)
6: Transmit data $\hat{\xi}_i^{s+1}$ to all ship $j \in \mathcal{M}\backslash\{i\}$.
7: **end for**
8: Wait for the update of $\hat{\xi}_j^{s+1}$ from all neighboring ships.
9: $s := s+1$.
10: **end for**

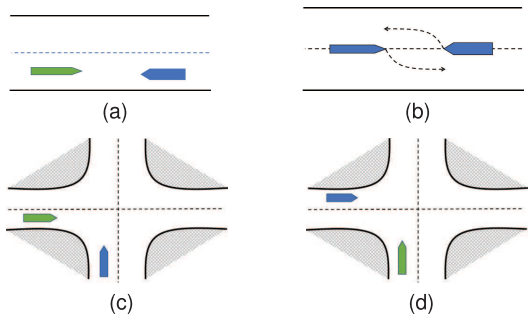


Fig. 3. Traffic situations according to The Netherlands' inland shipping police regulations. (a) and (b) Head-on situations. and (c) and (d) Crossing situations. The blue ship is the give-way ship, and the green is a stand-on ship.

B. Choosing $\alpha_{j,i}$ Based on Traffic Regulations

From (12), if we choose $\alpha_{j,i}$ large enough, then the cost for ship i to propose a course change for ship j will be prohibitive. Therefore, in this case, ship i will change its course instead of proposing a course change to j and behave as a give-way ship. Conversely, if we choose $\alpha_{j,i}$ small enough, ship i will behave as a stand-on ship by keeping its course and require ship j to change course. However, if $\alpha_{j,i}$ is neither too small nor too large, e.g., $\alpha_{j,i} = 1$, then behavior between ship i and j is decided through the negotiation process, i.e., the update iteration of NADMM. The stand-on or give-way priority between ship i and j is determined based on traffic regulations. This article uses traffic regulations adopted in Chapter 6 of The Netherlands' inland shipping police regulations [32]. The following rules are considered.

1) *Head-On Situation:* If two vessels are approaching each other on opposite courses in such a way that there is a risk of collision, the vessel not following the starboard side of the fairway shall give way to the vessel following the starboard side of the fairway [see Fig. 3(a)]. If neither vessel follows the starboard side of the fairway, each shall give way to vessels on the starboard side so that they pass each other port to port [see Fig. 3(b)].
2) *Crossing Situation:* If the courses of two ships cross each other in such a way that there is a risk of collision, the vessel not following the starboard side of the fairway

shall give way to the vessel following the starboard side of the fairway [see Fig. 3(c)]. In case none of the ships follows the starboard side of the fairway, the ship approaching from the port side gives way to the vessel approaching from starboard [see Fig. 3(d)].

3) *Overtaking Situation:* A vessel overtaking another vessel should keep out of the way of the overtaken vessel.

Consequently, we choose $\alpha_{j,i}$ as follows.

1) If ship i shall stand on to ship j, then $\alpha_{j,i} = K_{\mathrm{SO}}$, with $K_{\mathrm{SO}} \in (0, 1)$ small enough.
2) If neither of ship i nor j has stand on priority over each other then $\alpha_{j,i} = 1$.
3) If ship i shall give way to ship j, then $w_{ji} = K_{\mathrm{GW}}$, with $K_{\mathrm{GW}} > 1$ large enough.

Remark 1: The different values of $\alpha_{j,i}$ could change the outcome of the solution provided by Sync-CCAS. Therefore, $\alpha_{j,i}$ is determined before the NADMM start and kept unchanged during the negotiation process, even if there is a change in the priority between ships.

C. Convergence Analysis of Algorithm 1

First, let us rewrite the CCAS problem (13) in the following standard form:

$$\min_{w,v} F(w,v) = \frac{1}{M}\sum_{i\in\mathcal{M}} f_i(w_i) + g(v) \tag{15a}$$

$$\text{s.t. } A_i w_i - v - c_i = 0, \quad \text{in } \in \mathcal{M} \tag{15b}$$

where

$$w_i = \left[\tilde{u}_i^\top, \tilde{p}_i^\top\right]^\top \tag{16a}$$

$$v = \xi \tag{16b}$$

$$f_i(w_i) = \mathcal{J}_i(\tilde{p}_i, \tilde{u}_i) + \mathcal{G}_i(\tilde{p}_i, \tilde{u}_i) \tag{16c}$$

$$g(\xi) = 0 \tag{16d}$$

$$A_i = \left[0_{3M(N+1)\times 2MN} \quad \tilde{R}_i^{-1}\right] \tag{16e}$$

$$c_i = -\bar{\eta}_i^{\mathrm{WP}} \tag{16f}$$

and $\mathcal{G}_i(\tilde{p}_i, \tilde{u}_i)$ is an indicator function of the feasible region G_i, i.e., $\mathcal{G}_i(\tilde{p}_i, \tilde{u}_i) = 0$ if $(\tilde{p}_i, \tilde{u}_i) \in G_i$ and $\mathcal{G}_i(\tilde{p}_i, \tilde{u}_i) = \infty$ otherwise.

It is worth noting that if we combine the update (14e) and (14a) as follows:

$$\xi^{s+1} = \frac{1}{M}\sum_{j=1}^{M}\left\{\tilde{R}_i^{-1}\tilde{p}_i^{s+1} + \bar{\eta}_i^{\mathrm{WP}} + \frac{1}{\beta}z_i^{s+1}\right\}$$

and let

$$w = \left[w_1^\top, w_2^\top, \ldots, w_M^\top\right]^\top$$

$$z = \left[z_1^\top, z_2^\top, \ldots, z_M^\top\right]^\top$$

$$f(w) = \frac{1}{M}\sum_{i\in\mathcal{M}} f_i(w_i)$$

$$A = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_M \end{bmatrix}$$

$$B = \left[I_{3MN} \quad I_{3MN} \quad \cdots \quad I_{3MN}\right]^\top$$

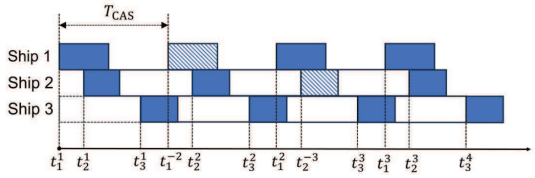


Fig. 4. Example of the synchronous update scheme.

then the NADMM update (14) becomes

$$z^{s+1/2} = z^s - \beta(1-\lambda)(Aw^s - c - Bv^s) \tag{17a}$$

$$w^{s+1} = \underset{w}{\mathrm{argmin}}\left\{f(w) + \langle z^{s+1/2}, Aw - Bv^s - c\rangle + \frac{\beta}{2}\|Aw - Bv^s - c\|^2\right\} \tag{17b}$$

$$z_i^{s+1} = z_i^{s+1/2} + \beta\left(Aw^{s+1} - c - Bv^s\right) \tag{17c}$$

$$v^{s+1} = \frac{1}{M}\sum_{j=1}^{M}\left\{A_i w_i^{s+1} - c_i + \frac{1}{\beta}z_i^{s+1}\right\} = \underset{v}{\mathrm{argmin}}\left\{\langle z^{s+1/2}, Aw^{s+1} - Bv - c\rangle + \frac{\beta}{2}\|Aw^{s+1} - Bv - c\|^2\right\}. \tag{17d}$$

Here, we get update (17b) by stacking the update (14c) of all M ships and taking into account that $\|Aw - Bv^s - c\|^2 = \sum_{i\in\mathcal{M}}\|A_i w_i - v^s - c_i\|^2$. Since (17) is identical to the NADMM update (9), the convergence of Algorithm 1 is equivalent to the convergence of the NADMM for problem (15).

Lemma 1: Consider the ADMM problem (15). The following statements hold for all $i \in \mathcal{M}$.

(i) f_i is proper and lsc, and g is proper, closed and convex.
(ii) A is surjective.
(iii) $(A_i f_i)$ is $L_{(A_i f_i)}$ – smooth, with constant $L_{(A_i f_i)} \in (0, +\infty)$. Hence (Af) is L – smooth, with $L = \max_{i\in\mathcal{M}}\{L_{(A_i f_i)}\}$.

Proof: See Appendix A.■

Lemma 1 shows that the problem (15) satisfies the requirement for the NADMM formulation [22, Asm. II]. Therefore, due to the convergence of NADMM, we have the convergence of Algorithm 1 as in Theorem 1.

Theorem 1 (Convergence of the Algorithm 1): Assume that problem (15) has a feasible solution and the communication between ships is synchronous. Let $\{w_i^s, z_i^s, \xi^s\}$ be generated by Algorithm 1 with $\lambda \in (0, 2)$, $\beta > 2L$, and $L > 0$. Then, the following holds for all $i \in \mathcal{M}$.

1) The residual $(Aw_i^s - \xi^s - c_i)_{k\in\mathbb{N}}$ vanishes with $\min_{i\le k}\|(Aw_i^s - \xi^s - c_i\| = o(1/\sqrt{k})$.
2) $\{w_i^s, z_i^s, \xi^s\}$ converges to a stationary point $\{w_i^*, z_i^*, \xi^*\}$.

IV. Asynchronous Distributed Collaborative Collision Avoidance Algorithm

Although the Sync-CCAS algorithm can guarantee a stationary solution for problem (13), it can be challenging to implement Sync-CCAS in real-time systems. Consider the following example with three ships with no coordinator (see Fig. 4). Suppose that three ships execute the Sync-CCAS

algorithm after every time interval T_{CAS}. Ships 1–3 start the algorithm at time t_1^1, t_2^1, and t_3^1, respectively. After the duration of T_{CAS} from the first iteration, ship 1 is activated again at time t_1^{-2}, but cannot proceed to the second iteration due to missing information from ship 3. Therefore, ship 1 must skip update at time t_1^{-2} and wait until the time t_1^2. As illustrated, it can take up to twice the interval time, i.e., $2T_{\text{CAS}}$, to proceed with one interaction of Sync-CCAS. This delay time causes ships to respond slower to collision risks and harms their safety.

This section presents an asynchronous version of the sync-CCAS algorithm, the Async-CCAS algorithm, to overcome the problem mentioned above. The Async-CCAS allows ships to update their decision without waiting for others, i.e., the asynchronous communication. To develop the Async-CCAS algorithm, we introduce the Async-NADMM.

A. Asynchronous NADMM

We derive the Async-NADMM algorithm from the asyn-FedDR algorithm proposed in [30]. Consider the following optimization problem:

$$\min_{u_i,\tau\in\mathbb{R}^p} \frac{1}{M}\sum_{i=1}^{M}\varphi_i(u_i)+\phi(\tau) \tag{18a}$$

$$\text{s.t. } u_i=\tau \quad \forall i\in\mathcal{M}. \tag{18b}$$

Here, φ_i is a nonconvex and L_{φ_i}-smooth extended-real-valued functions; ϕ is a proper, closed and convex extended-real-valued function. Then, a Douglas–Rachford iteration according to the asyncFedDR algorithm is as follows:

$$\begin{aligned}
\varsigma_i^{s+1} &= \varsigma_i^s+\lambda\left(u_i^s-\tau^s\right) \quad \forall i\in\mathcal{M}\\
u_i^{s+1} &= \mathbf{prox}_{\gamma\varphi_i}\left(\varsigma_i^{s+1}\right) \quad \forall i\in\mathcal{M}\\
\tau^{s+1} &= \mathbf{prox}_{\gamma\phi}\left[\frac{1}{M}\sum_{i=1}^{M}\left(2u_i^{s+1}-\varsigma_i^{s+1}\right)\right]
\end{aligned} \tag{19}$$

where s is the iteration index.

Based on the Async-FedDR, the Async-NADMM update for problem (15) is

$$z_i^{s+0.5} = z_i^s-\beta(1-\lambda)\left(A_iw_i^s-v-c_i^s\right) \quad \forall i\in\mathcal{M} \tag{20a}$$

$$w_i^{s+1} = \arg\min_{w_i}\left\{f_i(w_i)+\left\langle z_i^{s+0.5},A_iw_i-v^s-c_i\right\rangle + \frac{\beta}{2}\|A_iw_i+v^s-c_i\|^2\right\} \tag{20b}$$

$$\tilde{w}^{s+1} = \frac{1}{M}\sum_{i=1}^{M}\left(A_iw_i^{s+1}-c_i\right) \tag{20c}$$

$$z_i^{s+1} = z_i^{s+0.5}+\beta\left(A_iw_i^{s+1}-v^s-c_i\right) \quad \forall i\in\mathcal{M} \tag{20d}$$

$$v^{s+1} = \arg\min_{v}\left\{g(v)+\left\langle\frac{1}{M}\sum_{i=1}^{M}z_i^{s+1},\tilde{w}^{s+1}-v\right\rangle + \frac{\beta}{2}\|\tilde{w}^{s+1}-v\|^2\right\}. \tag{20e}$$

The algorithm of the Async-NADMM is presented in Algorithm 2. In this algorithm, a client can be seen as a controller that executes the algorithm, and the server can be one of the controllers that is designated to store the global variable. Additionally, if all controllers store their own version of the global variable and update its global variable's version whenever they receive an update from others, the update scheme is the same as that of Algorithm 1. The update scheme will be further discussed in Section IV-B.

Algorithm 2 Async-NADMM

Input: Initiate the server with $v=v^0$, and clients with $z_i=z_i^0$, $w_i=w_i^0$, for $i\in\mathcal{M}$.

1: **for** $s=1,\ldots,s_{\max}$ **do**
2: Choose client i_s from $\mathcal{M}$ with equal probability.
3: Client i_s receive update of $v^s(t-d_{i_s})$ from coordinator with a delay d_{i_s}.
4: [*local update*] Client i_s update local variables $w_{i_s}^{s+1}$, $z_{i_s}^{s+1}$ follow (20b), (20d), and send the result to coordinator.
5: [coordinator update] The server updates global variable v^s using (20e).
6: s:=s+1.
7: **end for**

Similar to the asyncFedDR algorithm, at each iteration s client i_s receive a copy of global variable after a time delay d_{i_s}, i.e., $v^s(t_s-d_{i_s})$, with the current time t_s. Each client performs the updates asynchronously without waiting for the others to finish. Furthermore, the following assumption is made.

Assumption 3: Assume that each client is activated at least once during $s_{\max}$ iterations of the Async-NADMM. Moreover, we assume that the time delays are bounded by $T_{\text{delay}}<\infty$, i.e., $0\le d_{i_s}\le T_{\text{delay}}$ for all i_s.

In order to show the convergence of Algorithm 2, we first show the universal equivalent between the Async-NADMM update (20) and the asyncFedDR update (19).

Theorem 2: Let (w_i,v,z_i) be generated by (20) with relaxation λ and penalty β large enough so that every ADMM minimization subproblem has a solution. Let

$$\begin{cases}\varsigma_i^s = A_iw_i^s-c_i-\dfrac{1}{\beta}z^s\\ u_i^s = A_iw_i^s-c_i\\ \tau^s = v^s\end{cases} \tag{21}$$

and

$$\begin{cases}\varphi_i = (A_if_i)(.+c_i)\\ \phi = g\\ \gamma=\dfrac{1}{\beta}.\end{cases} \tag{22}$$

Then, we have the following relation:

$$\begin{cases}\varsigma_i^{s+1} = \varsigma_i^s+\lambda\left(u_i^s-\tau^s\right) \quad \forall i\in\mathcal{M}\\ u_i^{s+1} = \mathbf{prox}_{\gamma\varphi_i}\left(\varsigma_i^{s+1}\right) \quad \forall i\in\mathcal{M}\\ \tau^{s+1} = \mathbf{prox}_{\gamma\phi}\left\{\dfrac{1}{M}\displaystyle\sum_{i=1}^{M}\left(2u_i^{s+1}-\varsigma_i^{s+1}\right)\right\}.\end{cases} \tag{23}$$

Moreover, if A_i is surjective for all $i\in\mathcal{M}$, and $\beta>L$, then it holds that

$$V_\gamma^s=\mathcal{L}_\beta^s \tag{24}$$

where

$$V_\gamma^s(\tau^s) = \phi(\tau^s) + \frac{1}{M}\sum_{i=1}^{M}\left[\varphi_i\left(u_i^s\right) + \left\langle \nabla\varphi_i\left(u_i^s\right), \tau^s - u_i^s\right\rangle + \frac{1}{2\gamma}\left\|\tau^s - u_i^s\right\|^2\right]$$

$$\mathcal{L}_\beta^s = g(v^s) + \frac{1}{M}\sum_{i=1}^{M}\left[f_i\left(w_i^s\right) + \left\langle z_i^s, A_i w_i^s - v^s - c_i\right\rangle + \frac{\beta}{2}\left\|A_i w_i^s - v^s - c_i\right\|^2\right].$$

Proof: See Appendix B■

Theorem 2 allows us to employ the convergence analysis of the asyncFedDR to the Async-NADMM.

Theorem 3: Suppose that Problem (15) has a feasible solution with $F^* := \inf F(w, v) > -\infty$ and Assumption 3 holds. Let (w_i, v, z_i) be generated by (20) with relaxation λ, penalty $\beta > L$. Then, the following conditions hold.

(i) (w^s, v^s) converges to (w^*, v^*) such that

$$\mathbb{E}\left[\|\nabla F(w^*, v^*)\|^2\right] < \epsilon^2 \tag{25}$$

where the expectation is taken with respect to all random variables and communication delays in the Async-ADMM algorithm, and ϵ is an arbitrary positive number.

(ii) The residual $A_i w_i^s - v^s - c_i$ is mean-square convergent to 0 with the rate $\min_{p\le s}\mathbb{E}[\|A_i w_i^p - v^p - c_i\|^2] = o(1/s)$.

Proof: Theorem 3(i) is a direct application of [30, Th. 4.1].

Theorem 3(ii). Lemma B.3 and the proof of [30, Th. 4.1] imply that

$$\sum_{s\in\mathbb{R}}\sum_{i=1}^{M}\mathbb{E}\left(\left\|\tau^s - u_i^s\right\|^2\right) \le D\left[F\left(w^0, v^0\right) - F^*\right] \tag{26}$$

where D is a positive constant defined as in [30, Lemma B.3]. Using (21), then (26) is equivalent to

$$\sum_{s\in\mathbb{R}}\sum_{i=1}^{M}\mathbb{E}\left(\left\|A_i w_i^s - v^s - c_i\right\|^2\right) \le D\left[F\left(w^0, v^0\right) - F^*\right]. \tag{27}$$

Since $F^* > -\infty$ and $F(w^0, v^0) \in \mathbb{R}$, then $(A_i w_i^s - v^s - c_i)_{s\in\mathbb{N}}$ is mean square convergent to 0 with the claimed rate.■

In Theorem 3(i), random variables exist due to step 2 of Algorithm 2, where we randomly chose a client i_s to perform the updates.

B. Asynchronous Distributed Collision Avoidance Algorithm

We introduce the asynchronous update scheme as in Algorithm 3. Unlike Algorithm 1, each ship does not need to wait for others to finish their update to proceed to the next iteration. Due to this asynchronous scheme, the global variables that each ship uses may differ. This can be illustrated in the following example (see Fig. 5). Suppose that we have three ships with different starting times. Besides, each ship needs a different time to perform the updates (the blue area in the timeline). At time $t = t_1^1$ and $t = t_2^1$, ships 1 and 2 perform global updates using $[\hat{\xi}_1^0, \hat{\xi}_2^0, \hat{\xi}_3^0]$. Then, at time $t = t_3^1$, ship 3 performs global update using $[\hat{\xi}_1^1, \hat{\xi}_2^1, \hat{\xi}_3^0]$. At the second iteration ($s = 2$), we also see that ship 1 performs the global update with $[\hat{\xi}_1^1, \hat{\xi}_2^1, \hat{\xi}_3^0]$. However, ship 2, at $t = t_2^2$, performs update with $[\hat{\xi}_1^1, \hat{\xi}_2^1, \hat{\xi}_3^1]$. As can be seen, each ship uses a different set of $\hat{\xi}_j^s$ when performing the global update, and this situation is considered by AsyncFedDR. Therefore, the Async-ADMM, an application of AsyncFedDR, was utilized to construct Algorithm 3.

Algorithm 3 Async-CCAS

for $s = 1, ..., s_{\max}$ **do**
  for all $i \in \mathcal{M}$ each ship computes in parallel **do**
    Receive $\hat{\xi}_j^s$ from neighboring ships, $j \in \mathcal{M}\backslash\{i\}$
    if $\hat{\xi}_j^s$ is unavailable **then**
      $\hat{\xi}_j^s = \hat{\xi}_j^{s-1}$
    end if
    Update the global variable ξ^{s+1} using (14a)
    Update local variables z_i^{s+1}, $\tilde{u}_i^{s+1}$, $\tilde{p}_i^{s+1}$ and $\hat{\xi}_j^{s+1}$ using (14b)–(14e)
    Transmit data $\hat{\xi}_i^s$ to all ship $j \in \mathcal{M}\backslash\{i\}$.
    $s := s + 1$.
  end for
end for

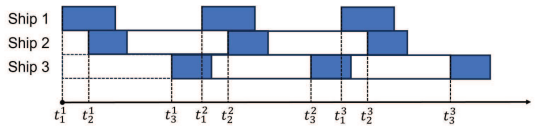


Fig. 5. Example of the asynchronous update scheme.

Given that each ship performs the updates after a constant time period, ΔT_{CAS}, and assuming that the calculation time is less than ΔT_{CAS}, the maximum delay time of the global variable T_{delay} is bounded by ΔT_{CAS} plus the data transmission time T_{trans}, i.e., $T_{\text{delay}} < \Delta T_{\text{CAS}} + T_{\text{trans}}$. If we further assume that the data transmission time is bounded, then Algorithm 3 satisfies Assumption 3. Therefore, the convergence of Algorithm 3 is guaranteed following Theorem 3.

V. Experimental Results

This section presents the evaluation of the performance of the proposed CCAS algorithms in both simulation and field experiments, where we consider that the ship sails along inland waterways. Therefore, the lateral control command $u_{i,i}^y$ is bounded by the width of the waterway. Although this, in practice, will depend on the lateral position of the ship, we represent this here with a constant symmetric constraint $|u_{i,i}^y| \le y_{\max}$. The predicted control signal for neighboring ships is also bounded by the maximum course angle change that the ship can make during the update interval ΔT, i.e., $0 \le \chi_{j,i}^d(k) \le \Delta\chi^{\max}$. The control parameters are presented in Table I. The parameters of the collision risk function are presented in Table II. In both simulation and field experiments, $K_{\text{SO}} = 0.12$ and $K_{\text{GW}} = 10^6$. Moreover, the Casadi toolbox [33] with the Interior Point Optimizer—ipopt [34]—is used to solve the local MPC problem (14c).

A. Performance Evaluation

We use two different ship models for simulation and field experiments. In field experiments, ships with dimensions of

TABLE I
CONTROL PARAMETERS

Parameter	Unit	Simulation	Field
$y_{\max}$	m	60	10
$\chi^{\max}$	rad	$\pi/6$	$\pi/6$
T_1/T_j	-	28.458	12.82
K_e	-	0.01	1/15
K_y	–	10^{-2}	2×10^{-2}
K_u	–	2×10^{-2}	4×10^{-2}
β	–	3×10^{-4}	5×10^{-4}
ΔT	s	20	5

TABLE II
COLLISION RISK FUNCTION'S PARAMETERS

Parameter	Scenario	Unit	Simulation	Field
α_{xj}	Head on & Overtaking	m	80	5.0
	Crossing		55	5.0
α_{yj}	Head on & Overtaking	m	25	1.2
	Crossing		50	2.0
K_{ca}	Head on & Overtaking	-	25	25
	Crossing		400	30
K_{d}	Head on	-	5	3
	Crossing		0	1
	Overtaking		0	0

2×1.08m are used. In simulations, a ship model has a length of 51.5m, and a beam of 8.6m is used. Due to the significance of the size and shape of the two models, we adopt two methods to evaluate the safety performance of the CCAS algorithms.

In the field experiments, we define a threshold distance of 4m for the safety distance of the ship. Their navigation is safe if the distance d_{ij} between ships i and j is greater or equal to the threshold.

Due to the ship's dimensions, it is not practical to use the distance (from the center of mass) between two ships in simulations to evaluate the safety of navigation. For example, the safety distance between two ships in a head-on situation can be as small as 16 m. For this reason, we define a rectangular safety domain around the ship so that if there is no other ship within this domain, then the navigation is safe. An index ϵ_i, with $\epsilon_i \leq 0$, indicates that there is another ship in the safety domain, which is defined as follows:

$$\epsilon_i = \min_{j\in\mathcal{M}\setminus\{i\}} \left\{\max\left\{|x_{i,i}-x_{j,i}|-d_i^x, |y_{i,i}-y_{j,i}|-d_i^y\right\}\right\} \tag{28}$$

where $2d_i^x$ and $2d_i^y$ are the length and width of the rectangle safety area, respectively. In this simulation, $d_i^x = 51.5$ m and $d_i^y = 8.6$ m, and unit for ϵ_i is meter. From (28), if there is a ship j with the lateral distance (in x-axis) and vertical distance (in y-axis) both less than the threshold then $\epsilon_i < 0$. While $\epsilon_i > 0$ indicates a collision-free navigation, the larger value of ϵ_i means the more distance to other ship.

Besides the safety performance, the computational performance is evaluated based on the computational time that the CCAS algorithms need to perform an update. The computational time is acceptable if its maximum is less than the update interval ΔT (see Table I).

B. Simulation Experiments

The simulation experiments are done in MATLAB 2024a running on a PC with an Intel[1] Core[2] i7-11850H and 32 GB of RAM. The CCAS algorithm is called every 20 s.

The purposes of the simulation experiments are to illustrate the scalability of the Sync-CCAS algorithm and compare it to the Async-CCAS algorithm. Therefore, only the scenarios of four and six ships are presented. We also present the computational time with each ship's maximum and mean time in Table III.

In the simulation environment, we use a shared variable to store a copy of all the decisions of the ships, i.e., ξ_i. Each ship does not receive updates directly from ξ_i, but from the copy that is stored in the share variable. Depending on the time the shared variable is updated, we can simulate synchronous or asynchronous communication. To simulate synchronous communication, we update variables ξ_i of all M ships simultaneously after they have all finished their individual update. This way guarantees that all ships receive the same information for each iteration update. On the contrary, in asynchronous experiments, we update the shared variable every time a ship i has updated its decision ξ_i. Due to multiple updates of the shared variable in one iteration, the ship that performs updates later can receive different inputs than the former. Although there is no actual delay in communication between ships in the simulation environment, thanks to the difference in inputs each ship receives, we can simulate asynchronous communication.

In the four-ship scenario with Sync-CCAS algorithm (Fig. 6), ships 1 and 4 have the stand-on priority but need to negotiate with each other to avoid a collision. On the other hand, ship 3 must give way to all others, while ship 2 must give way to ships 1 and 4. As shown in Fig. 6(b) and (c), the situation is solved according to the traffic rules.

In the four-ship scenario, we also compare the performance of the Sync-CCAS and Async-CCAS algorithms with different values of maximum iteration $s_{\max}$. As illustrated in Table III, the maximum and mean values of the computation time of an iteration are barely changed when $s_{\max}$ is increased. Furthermore, Fig. 7 shows the similar safety indices between $s_{\max} = 2$ and $s_{\max} = 5$. However, there is a slight decrease in safety indices in case $s_{\max} = 10$. Taking into account that an increase in the value of $s_{\max}$ would require more computation time (in total) but give no improvement in performance, we can conclude that $s_{\max} = 2$ is the optimal value for $s_{\max}$.

The six-ship scenario (Figs. 8 and 9) is more complex, where multiple ships have the stand-on priority, e.g., ships 1 and 4–6. Therefore, negotiation is required to resolve the situation. The maximum iteration in this scenario is $s_{\max} = 2$. Fig. 8 presents the result of the Sync-CCAS, while the result of the Async-CCAS is presented in Fig. 9. The ships with give-way obligations (ships 2 and 3) strictly follow the traffic rules. Ship 2 gave way to all others except ship 3, and ship 3 gave way to all others [Figs. 8(d) and 9(d)]. However, there is a difference in the behaviors between ships with stand-on priorities. In the synchronous communication scheme, ship 1

[1]Registered trademark.
[2]Trademarked.

TABLE III
COMPUTATIONAL TIME (IN SECONDS) NEEDED TO PERFORM AN UPDATE IN SIMULATIONS

Scenario		Ship 1	Ship 2	Ship 3	Ship 4	Ship 5	Ship 6
4 ships, Sync-CCAS, $s_{\max} = 2$ (Fig. 6)	Max	3.309	2.945	2.671	3.430	-	-
	Mean	2.259	1.633	1.137	1.971	-	-
4 ships, Async-CCAS, $s_{\max} = 2$	Max	2.678	2.927	2.720	2.803	-	-
	Mean	1.896	1.559	1.268	1.765	-	-
4 ships, Async-CCAS, $s_{\max} = 5$	Max	2.724	2.674	2.710	2.943	-	-
	Mean	1.789	1.584	1.355	1.636	-	-
4 ships, Async-CCAS, $s_{\max} = 10$	Max	3.027	3.148	2.738	3.093	-	-
	Mean	1.793	1.566	1.391	1.914	-	-
6 ships, Sync-CCAS, $s_{\max} = 2$ (Fig. 8)	Max	6.567	4.289	5.190	5.694	4.889	5.165
	Mean	3.683	2.52	2.446	3.474	3.065	3.227
6 ships, Async-CCAS, $s_{\max} = 2$ (Fig. 9)	Max	4.900	4.118	7.631	5.577	4.089	4.067
	Mean	2.798	1.984	2.551	2.697	2.834	2.519
6 ships, Async-CCAS, $s_{\max} = 2$, data loss 5%	Max	5.453	5.738	4.638	5.699	5.837	6.708
	Mean	3.503	2.575	2.574	3.506	3.218	3.333

$s_{\max}$: maximum iteration of CCAS algorithms.

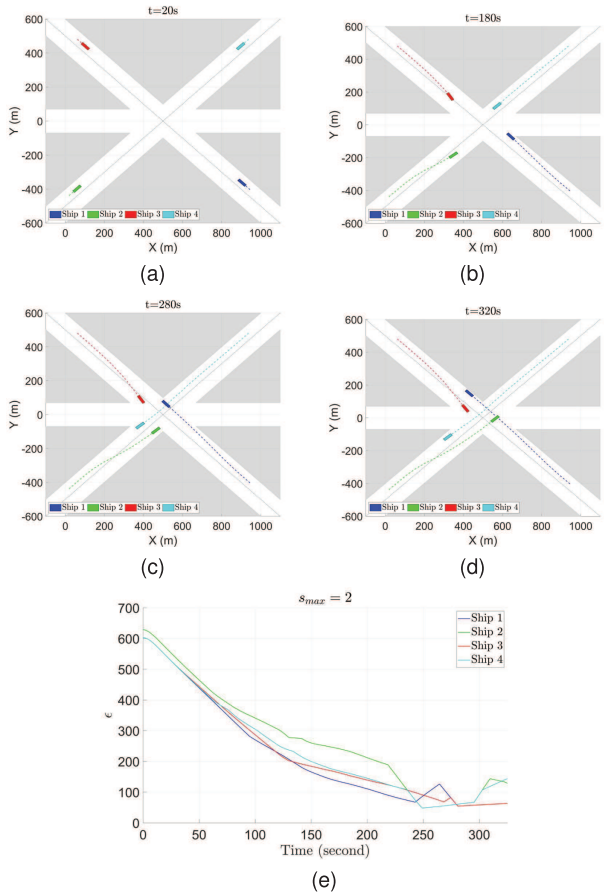


Fig. 6. Simulation experiments (synchronous communication): intersection crossing between four ships. The guiding lines are depicted in the black dashed lines. (a)–(d) Trajectories of ships at different times. (e) Safety indices of ships.

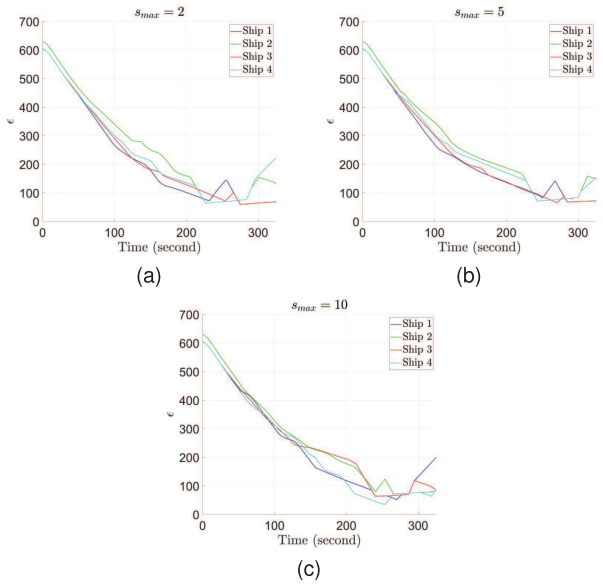


Fig. 7. Safety indices of four-ship scenarios with different values of maximum iteration of CCAS algorithm $s_{\max}$. (a) $s_{\max} = 2$. (b) $s_{\max} = 5$. (c) $s_{\max} = 10$.

gave way to ship 4 [Fig. 8(c)], while in the asynchronous communication scheme, ship 4 gave way to ship 1 [Fig. 9(c)]. This is because the give-way or stand-on action between ships with the same stand-on priorities is up to negotiation, and the negotiation result is influenced by the communication scheme. It is also shown that the order in which ships update their intentions can affect the negotiation outcome between ships with the same stand-on priorities.

Although both Sync-CCAS and Async-CCAS algorithms can resolve the situation without any significant collision risk, the computation time is increased compared to the four-ship scenarios (see Table III). However, these calculation times are acceptable. It is worth noting that the average computational times of the Sync-CCAS and Async-CCAS algorithms are similar.

We further evaluate the performance of the Async-CCAS in the condition of unstable communication between ships. The traffic situation setup is the same as in Fig. 8. However, each ship i has 5% that its update $\hat{\xi}_i^s$ cannot reach other neighboring ships, e.g., ship i lost connection for a short time. Due to the data loss, both the maximum and average computation time

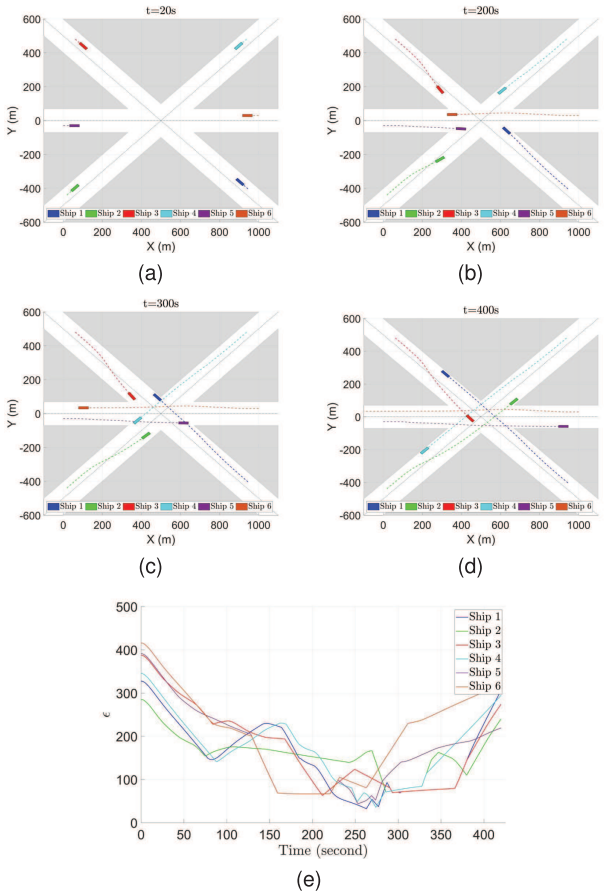


Fig. 8. Simulation experiments (synchronous communication): intersection crossing between six ships. The guiding lines are depicted in the black dashed lines. (a)–(d) Trajectories of ships at different times. (e) Safety indices of ships.

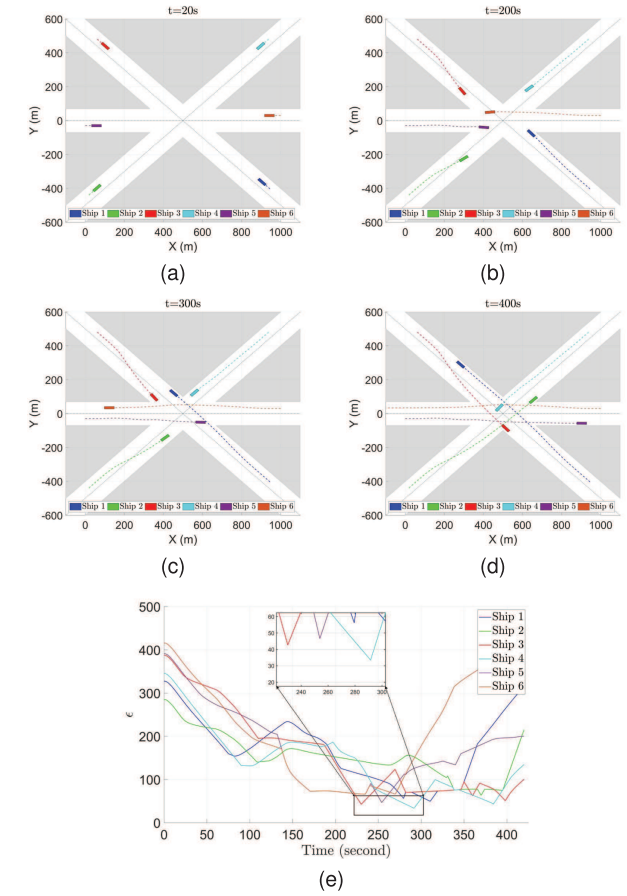


Fig. 9. Simulation experiments (asynchronous communication): intersection crossing between six ships. The guiding lines are depicted in the black dashed lines. (a)–(d) Trajectories of ships at different times. (e) Safety indices of ships.

increase (see Table III). Moreover, the safety performance in case of data loss is also decreased (Fig. 10) compared to the case that has no data loss [Fig. 9(e)].

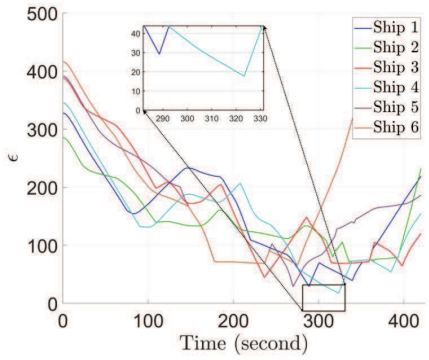


Fig. 10. Safety indices in case of 5% data loss.

C. Field Experiments

We verify the performance of the Async-CCAS algorithm with field experiments in the harbor area near Nyhavna, Trondheim, Norway. Three autonomous surface vessels (ASVs) called NTNU Fish Otters (see Fig. 11) are used in the field experiments. Each NTNU Fish Otter is equipped with a computation unit with the core of the Raspberry Pi CM4108032. Additionally, the ASVs communicate with each other through a 4G connection. Further details of the hardware information of the NTNU Fish Otters can be found in [35].

In field experiments, the Async-CCAS algorithm is designed to run on the LSTS toolchain [36] and is executed by the onboard computer. The update signal $\hat{\xi}_i^s$ is exchanged between ships through the IMC communication protocol. Each ASV performs the local update every 5 s, with the maximum iteration $s_{\max} = 2$. To reduce the computation time, we generate a customized solver for the problem (14) using the Casadi codegen tool. Then, the parameters, i.e., the initial state $p_{j,i}^{\text{init}}$, are passed directly to the solver.

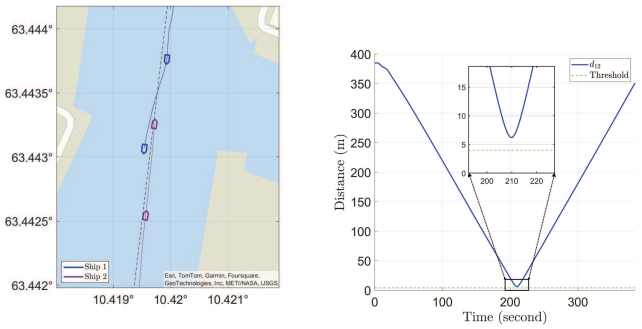
The computation time of each ship is present in Table IV. For comparison, we also present in Table IV the computation time from simulations run in the LSTS toolchain with the same scenarios but with Intel[1] Core[2] i7-11850H hardware and 8 GB of RAM. The visual results from simulations are not presented due to the similarity between them and the field experiments. However, we present a case where there is a significant difference between simulation and field experiment results.

1) Two-Ship Scenarios: In the first part of the experiment, we verify the performance of Algorithm 3 in scenarios with two ships, which are prevalent traffic scenarios. The results of head-on, crossing, and overtaking between two ships are

TABLE IV
COMPUTATIONAL TIME (IN SECONDS) NEEDED TO PERFORM AN UPDATE IN FIELD EXPERIMENTS COMPARED TO SIMULATION

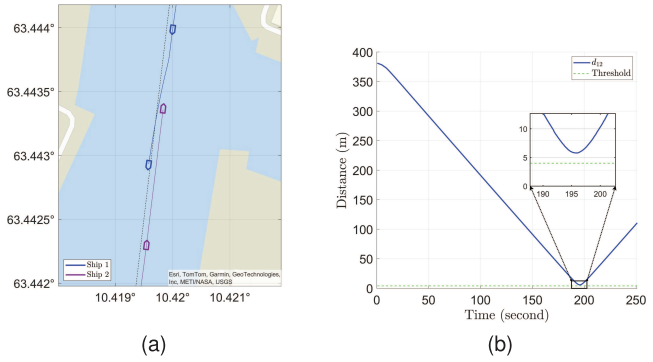
Scenario		Field experiment			Simulation		
		Ship 1	Ship 2	Ship 3	Ship 1	Ship 2	Ship 3
Head-on (Fig. 12 & Fig. 13)	Max	1.725	1.128	-	0.308	0.422	-
	Mean	1.002	0.711	-	0.169	0.255	-
Crossing (Fig. 14)	Max	1.590	0.427	-	0.279	0.210	-
	Mean	0.867	0.211	-	0.176	0.058	-
Overtaking (Fig. 15)	Max	1.046	0.846	-	0.283	0.260	-
	Mean	0.518	0.400	-	0.125	0.098	-
Combined case 1 (Fig. 16)	Max	1.445	2.143	1.413	0.217	0.671	0.108
	Mean	0.997	1.441	0.457	0.102	0.290	0.035
Combined case 2 (Fig. 17)	Max	0.873	2.546	2.443	0.159	0.560	0.515
	Mean	0.422	1.314	1.509	0.047	0.313	0.279

Fig. 11. NTNU Fish Otter.



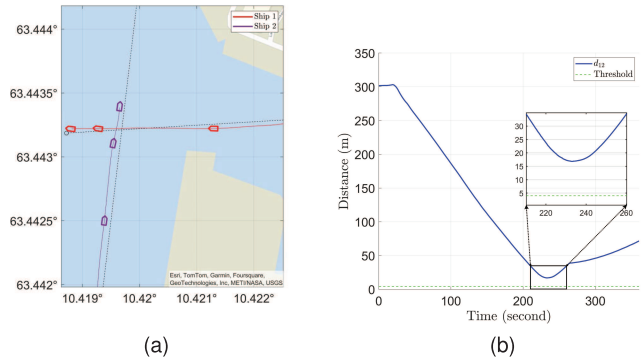
(a) (b)

Fig. 12. Field experiments: head-on between two ships. Ship 2 has the stand-on priority. The guiding lines are depicted in the black dashed lines. (a) Trajectories of two ships. (b) Distance between two ships.



(a) (b)

Fig. 13. Simulation: head-on between two ships. Ship 2 has the stand-on priority. The guiding lines are depicted in the black dashed lines. (a) Trajectories of two ships. (b) Distance between two ships.



(a) (b)

Fig. 14. Field experiments: intersection crossing between two ships. Ship 1 has the stand-on priority. The guiding lines are depicted in the black dashed lines. (a) Trajectories of two ships. (b) Distance between two ships.

shown in Figs. 12–15, respectively. In three scenarios, both stand-on and give-way ships behaved appropriately. In each scenario, the give-way ship changed course (head-on and overtaking scenarios) or reduced speed (crossing scenario) to give way to the stand-on ship.

As shown in Figs. 12 and 13, there is a difference between the trajectory of ship 2 (the stand-on ship) in the field experiment compared to the simulation result. Due to the strong wind on the experiment day, ship 2 drifted toward the port side. In response to the drift of ship 2, ship 1 made a larger course change toward the starboard side.

2) Three-Ship Scenarios: The second part of the experiment involves three ships in combined head-on and crossing scenarios (see Figs. 16 and 17). In both scenarios, ships 1 and 2 encounter each other head-on, and simultaneously, ship 3 passes by from the east. The only difference between the two scenarios is that in the first one (Fig. 16), there is a clear stand-on and give-way order between the three ships.

In the first scenario [Fig. 16(a)], ship 3 must give way for ships 1 and 2 because it is sailing on the port sail of the guiding line. Ship 1 must give way to ship 2, and ship 2 can stand on. With a clear priority order, Algorithm 3 resolved the situation as shown in Fig. 16(c) and (d).

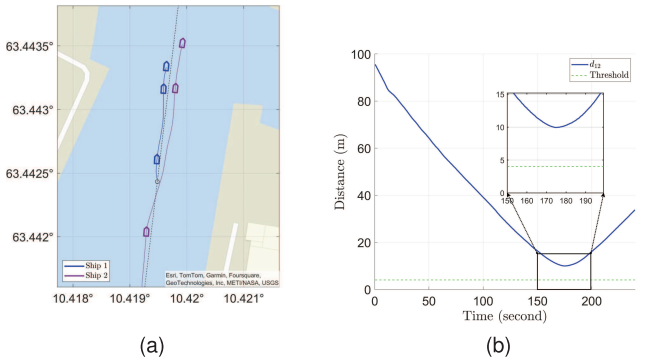


Fig. 15. Field experiments: overtaking between two ships. Ship 1 was overtaken by ship 2. The guiding lines are depicted in the black dashed lines. (a) Trajectories of two ships. (b) Distance between two ships.

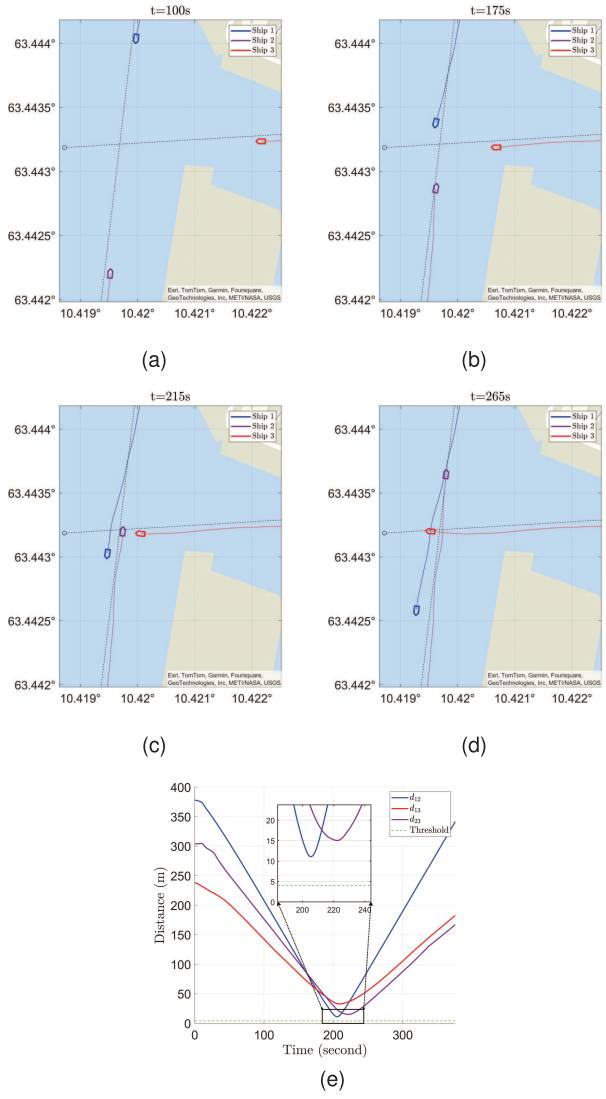


Fig. 16. Field experiments: intersection crossing between three ships (case 1). The guiding lines are depicted in the black dashed lines. (a)–(d) Trajectories of ships at different times. (e) Distance between ships.

In the second scenario [Fig. 17(a)], ships 2 and 3 sail on the starboard side and, therefore, can stand on. In this scenario, only ship 1 must give way to ships 2 and 3. This is a situation

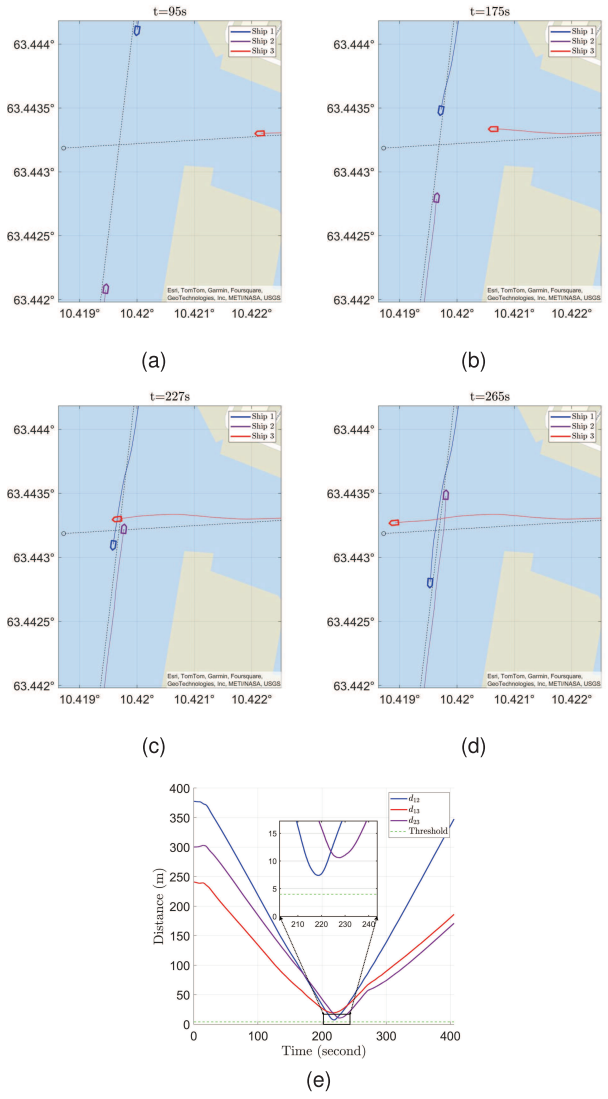


Fig. 17. Field experiments: intersection crossing between three ships (case 2). The guiding lines are depicted in the black dashed lines. (a)–(d) Trajectories of ships at different times. (e) Distance between ships.

where the priority by traffic rules is unclear since ships 2 and 3 both have stand-on priority. Therefore, a negotiation must be carried out between ships 2 and 3. As illustrated in Fig. 17(e), the minimum distance between ships 2 and 3 is approximately 10 m and significantly lower than that of the first scenario, which is 15 m [see Fig. 16(e)]. Additionally, from Table IV, the mean and maximum calculating time of ship 3 in Fig. 16 is higher than in Fig. 17. The increase in calculating time and decrease in distance is due to the unclear priority between ships 2 and 3, which require more negotiation time. Therefore, evasive actions were executed later. Nevertheless, the minimum distances in both scenarios are above the safety threshold.

VI. Discussion

A. Robustness Under Environment Uncertainties

The proposed CCAS algorithms do not directly consider the environmental uncertainties because they use a kinematic model instead of a dynamic model. The low-level controller

handles the environmental uncertainties and is assumed to be capable of achieving the command from the CCAS algorithms. Typically, environmental uncertainties can be compensated by the low-level controller. In case the kinematic models are inaccurate, the solution is to increase safety parameters, i.e., α_{xj}, α_{yj}.

However, even with an accurate model, the ship can drift from the planned trajectories due to the influence of external disturbance, e.g., strong wind or current, as shown in Figs. 12 and 13. When the ship drifted, the actual trajectory of the ship differed from the planned trajectory that was sent to others, i.e., $\hat{\xi}_i$, and could cause a collision. To overcome this problem, at every last iteration of the Async-CCAS algorithm, i.e., $s = s_{\max}$, we replace the planned trajectory of the own-ship (only a part of $\hat{\xi}_i$) that is provided by the Async-CCAS algorithm with the predicted trajectory from a prediction algorithm that uses a hydrodynamic model. This replacement may impact CCAS algorithms, so it is done only in the last iteration to minimize effects.

B. Scalability of the Async-CCAS

As shown in Tables III and IV, the computation time increases when the number of participants increases. However, Table IV shows that the computational time in simulation, which used better computational hardware, is significantly lower than in the field experiment. It suggests that with sufficient computation capacity, the Async-CCAS is scalable. Furthermore, a better optimization solver can also help reduce the computational time.

It is also important to note that the set $\mathcal{M}$ only considers ships that involve in a traffic situation and excludes others, e.g., ships that are moored, too far away, have been passed, or in a different (nearby) waterway without collision risk. A practical implementation would need to define the set $\mathcal{M}$. Additionally, it is important to keep M as small as possible due to computational complexity.

C. Mixed Traffic Scenarios

In this study, we only consider the traffic scenario that involves uncrewed ships. However, in actual traffic situations, an uncrewed ship may encounter both crewed and uncrewed ships [37]. We can expect a crewed ship j to share its own predicted trajectory, but it cannot exchange the whole solution, i.e., $\hat{\xi}_j$. One possible solution is to predict the missing information of $\hat{\xi}_j$ based on the shared predicted trajectory and the traffic situation. However, further investigation is needed to find a comprehensive solution.

D. Performance Evaluation

In this article, we used two criteria to evaluate the performance of the CCAS algorithms: the computational time and the safety index. While these two criteria can show us the algorithm's practicality and the safety of the proposed solutions, they may not show the difference in performance between our proposed algorithm and others. There is a lack of a benchmark framework to evaluate and compare different CCAS algorithms' performance systematically.

VII. Conclusion and Future Research

This article presented two distributed MPC-based ADMM algorithms to solve the problem of CCAS for autonomous ships in inland waterways. The proposed algorithms allow ships to collaborate to avoid collisions and comply with inland traffic regulations. On the one hand, the synchronous algorithm is designed for an ideal condition in which the information exchange process is synchronized between ships. On the other hand, the asynchronous algorithm is designed to handle cases where synchronized communication is not available. We overcome the obstacle of asynchronous communication by extending the result of the asyncFedDR.

The simulation results show that the synchronous algorithm can guarantee collision-free navigation for ships in complex scenarios. In the simulation environment, the synchronous and asynchronous algorithms show similar results and computational times. Both algorithms comply with traffic rules in the tested scenarios. Furthermore, field experiments confirm the safety and computational performance of the asynchronous algorithm under practical conditions. However, the simulation results suggest that the proposed algorithms' performance can decrease in more extreme conditions, such as unstable communication between ships.

As discussed in Section VI, there are points that need to be improved to allow the Async-CCAS to work in inland waterway traffic. In future research, we will focus on developing a protocol to support the participation of human-operated ships in the CCAS framework. Additionally, the computational time of the Async-CCAS algorithm could increase when handling the cases where the give-way or stand-on role of each ship is not clearly decided by traffic rules. Future research could aim to reduce computation time in these cases.

Appendix A
Proof of Lemma 1

Lemma 1(i): $f_i(w_i)$ is a sum of an indicator function of a closed set $\mathcal{G}_i(\tilde{p}_i, \tilde{u}_i)$ and a continuous function $\mathcal{J}_i(\tilde{p}_i, \tilde{u}_i)$. Thus, $f_i(w_i)$ is lsc. Since $g(\xi) = 0$, then g is proper, closed, and convex.

Lemma 1(ii): A_i is a full-row rank matrix and is then surjective. Hence, A is surjective.

Lemma 1(iii): First, we show that there exists $L_{A_i f_i} > 0$ such that

$$-L_{A_i f_i} \left\| A_i \left(w_i^1 - w_i^2 \right) \right\|^2 \leq \left\langle \nabla f_i \left(w_i^1 \right) - \nabla f_i \left(w_i^2 \right), w_i^1 - w_i^2 \right\rangle \leq L_{A_i f_i} \left\| A_i \left(w_i^1 - w_i^2 \right) \right\|^2 \tag{29}$$

whenever $\nabla f_i(w_i^1), \nabla f_i(w_i^1) \in \mathbf{range}(A_i^\top)$. Then, $\nabla f_i(w_i) \in \mathbf{range}(A_i^\top)$ is equivalent to $\tilde{u}_i = 0$ and $\mathcal{G}_i = 0$. In that case, $f_i(w_i) = \mathcal{J}_i(\tilde{p}_i, 0) = \mathcal{J}_i^{\text{ca}}(\tilde{p}_i)$ is the Lipschitz continuous gradient function due to $R_{ij}(\cdot)$ being L_R-smooth. Then, there exists L_{f_i} such that

$$\left\| \nabla f_i \left(w_i^1 \right) - \nabla f_i \left(w_i^2 \right) \right\| \leq L_{f_i} \left\| w_i^1 - w_i^2 \right\|.$$

With the help of the Cauchy–Schwartz inequality, we have

$$L_{f_i} \left\| w_i^1 - w_i^2 \right\|^2 \geq \left\| \nabla f_i \left(w_i^1 \right) - \nabla f_i \left(w_i^2 \right) \right\| \cdot \left\| w_i^1 - w_i^2 \right\|$$

$$\geq \left|\left\langle \nabla f_i\left(w_i^1\right) - \nabla f_i\left(w_i^2\right), w_i^1 - w_i^2\right\rangle\right|$$

or equivalently

$$-L_{f_i}\left\|w_i^1 - w_i^2\right\|^2 \leq \left\langle \nabla f_i\left(w_i^1\right) - \nabla f_i\left(w_i^2\right), w_i^1 - w_i^2\right\rangle \leq L_{f_i}\left\|w_i^1 - w_i^2\right\|^2. \tag{30}$$

We recall fact that $\tilde{R}_i\tilde{p}_i - c_i = \xi$, and R_i, which is the component of $\tilde{R}_i$, is the rotation matrix. Therefore, $\|\tilde{p}_i^1 - \tilde{p}_i^2\|^2 = \|\xi^1 - \xi^2)\|^2$. Then, for all $\nabla f_i(w_i) \in \mathbf{range}(A_i^\top)$, we have

$$\begin{aligned} L_{f_i}\left\|w_i^1 - w_i^2\right\|^2 &= L_{f_i}\left\|\tilde{p}_i^1 - \tilde{p}_i^2\right\|^2 \\ &= L_{f_i}\left\|\xi^1 - \xi^2\right\|^2 \\ &= L_{f_i}\left\|A_i\left(w_i^1 - w_i^2\right)\right\|^2. \end{aligned} \tag{31}$$

From (30) and (31), we get that there exists $L_{A_if_i}$ that satisfies (29). Then, following [22, Th. 5.13], (A_if_i) is $L_(A_if_i)$-smooth. Furthermore, if we choose L such that $L = \max_{i\in\mathcal{M}}\{L_{(A_if_i)}\}$, we also have

$$-L\|A(w^1 - w^2)\|^2 \leq \langle \nabla f(w^1) - \nabla f(w^2), w^1 - w^2\rangle \leq L\|A(w^1 - w^2)\|^2$$

holds whenever $\nabla f(w^1), \nabla f(w^1) \in \mathbf{range}(A^\top)$. Therefore, (Af) is L-smooth.

Appendix B
Proof of Theorem 2

First, we show that (23) holds. Following (21), we have:

$$\begin{aligned} \varsigma_i^s + \lambda\left(\tau^s - u_i^s\right) &= A_iw_i^s - c_i - \frac{z_i^s}{\beta} - \lambda\left(A_iw_i^s - v^s - c_i\right) \\ &\overset{(20a)}{=} A_iw_i^s - c_i - \frac{z_i^{s+0.5}}{\beta} - \left(A_iw_i^s - v^s - c_i\right) \\ &\overset{(20d)}{=} A_iw_i^{s+1} - c_i - \frac{z_i^{s+1}}{\beta} = \varsigma_i^{s+1}. \end{aligned} \tag{32}$$

Moreover, from (32), we also observe that

$$\varsigma_i^{s+1} = -\frac{z_i^{s+0.5}}{\beta} - v^s. \tag{33}$$

Next, we have

$$\begin{aligned} &\mathbf{prox}_{\gamma\varphi_i}\left(\varsigma_i^{s+1}\right) \\ &= \arg\min_{\mu}\left\{(A_if_i)(\mu + c_i) + \frac{1}{2\gamma}\left\|\mu - \varsigma_i^{s+1}\right\|^2\right\} \\ &= \arg\min_{\mu}\left\{\inf_{w_i} f_i(w_i) + \frac{1}{2\gamma}\left\|\mu - \varsigma_i^{s+1}\right\|^2 \middle| A_iw_i = \mu + c_i\right\} \\ &= A_i\arg\min_{w_i}\left\{f_i(w_i) + \frac{1}{2\gamma}\left\|A_iw_i - c_i - \varsigma_i^{s+1}\right\|^2\right\} - c_i. \end{aligned} \tag{34}$$

Then, we obtain

$$\begin{aligned} u_i^{s+1} &= A_iw_i^{s+1} - c_i \\ &= A_i\arg\min_{w_i}\left\{f_i(w_i) + \left\langle z_i^{s+0.5}, A_iw_i - v^s - c_i\right\rangle \right. \\ &\qquad \left. + \frac{\beta}{2}\|A_iw_i + v^s - c_i\|^2\right\} - c_i \\ &= A_i\arg\min_{w_i}\left\{f_i(w_i) + \frac{\beta}{2}\left\|A_iw_i + v^s - c_i + \frac{z_i^{s+0.5}}{\beta}\right\|^2\right\} \\ &\qquad - c_i \\ &\overset{(33)}{=} A_i\arg\min_{w_i}\left\{f_i(w_i) + \frac{\beta}{2}\left\|A_iw_i - c_i - \varsigma_i^{s+1}\right\|^2\right\} - c_i \\ &\overset{(34)}{=} \mathbf{prox}_{\gamma\varphi_i}\left(\varsigma_i^{s+1}\right). \end{aligned}$$

Similarly

$$\begin{aligned} &\tau^{s+1} \\ &= v^{s+1} \\ &= \arg\min_{v}\left\{g(v) + \left\langle \frac{1}{M}\sum_{i=1}^{M} z_i^{s+1}, \tilde{w}^{s+1} - v\right\rangle \right. \\ &\qquad \left. + \frac{\beta}{2}\|\tilde{w}^{s+1} - v\|^2\right\} \\ &= \arg\min_{v}\left\{g(v) + \frac{\beta}{2}\left\|\frac{1}{M}\sum_{i=1}^{M}\left(A_iw_i^{s+1} - c_i + \frac{z_i^{s+1}}{\beta}\right) - v\right\|^2\right\} \\ &= \arg\min_{v}\left\{g(v) + \frac{\beta}{2}\left\|v - \frac{1}{M}\sum_{i=1}^{M}\left(2u_i^{s+1} - \varsigma_i^{s+1}\right)\right\|^2\right\} \\ &= \mathbf{prox}_{\gamma\phi}\left\{\frac{1}{M}\sum_{i=1}^{M}\left(2u_i^{s+1} - \varsigma_i^{s+1}\right)\right\}. \end{aligned}$$

Now, to prove (24), we first show that

$$\nabla\varphi_i\left(u_i^s\right) = -z_i^s \tag{35}$$

hold for all $i \in \mathcal{M}$. Indeed, since

$$\begin{aligned} u_i^s &= \mathbf{prox}_{\gamma\varphi_i}\left(\varsigma_i^s\right) \quad \forall i \in \mathcal{M} \\ z_i^s &= -\frac{\varsigma_i^s - u_i^s}{\gamma} \end{aligned}$$

and together with (2) imply that

$$z_i^s = -\frac{\varsigma_i^s - u_i^s}{\gamma} = -\nabla\varphi_i\left(u_i^s\right). \tag{36}$$

Next, [22, Prop 5.2(ii)] gives

$$\varphi\left(u_i^s\right) = (A_if_i)\left(A_iw_i^s\right) = f_i\left(w_i^s\right). \tag{37}$$

From (36) and (37), (24) holds.

Acknowledgment

The authors would like to thank Melih Akdağ for providing the code template that appears as a foundation for the collision avoidance algorithm in a field experiment. They would also like to express their gratitude to Daniel Bogen, Jo Arve Alfredsen, Lorenzo Govani, and Morten Einarsve for your invaluable support during the field experiment. This publication reflects only the authors' view, exempting European Union from any liability.

REFERENCES

[1] D. Menges, T. Tengesdal, and A. Rasheed, "Nonlinear model predictive control for enhanced navigation of autonomous surface vessels," *IFAC-PapersOnLine*, vol. 58, no. 18, pp. 296–302, 2024.

[2] D. Mahipala and T. A. Johansen, "Model predictive control for path following and collision-avoidance of autonomous ships in inland waterways," in *Proc. 31st Medit. Conf. Control Autom. (MED)*, Jun. 2023, pp. 896–903.

[3] T. A. Johansen, T. Perez, and A. Cristofaro, "Ship collision avoidance and COLREGS compliance using simulation-based control behavior selection with predictive hazard assessment," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 12, pp. 3407–3422, Dec. 2016.

[4] T. Trym, E. F. Brekke, and T. A. Johansen, "On collision risk assessment for autonomous ships using scenario-based MPC," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14509–14516, 2020.

[5] G. Zhang, Y. Wang, J. Liu, W. Cai, and H. Wang, "Collision-avoidance decision system for inland ships based on velocity obstacle algorithms," *J. Mar. Sci. Eng.*, vol. 10, no. 6, p. 814, Jun. 2022.

[6] S. V. Rothmund, T. Tengesdal, E. F. Brekke, and T. A. Johansen, "Intention modeling and inference for autonomous collision avoidance at sea," *Ocean Eng.*, vol. 266, Dec. 2022, Art. no. 113080.

[7] M. Akdağ, P. Solnør, and T. A. Johansen, "Collaborative collision avoidance for maritime autonomous surface ships: A review," *Ocean Eng.*, vol. 250, Apr. 2022, Art. no. 110920.

[8] C. Guiking. (2022). *Digital Intention Sharing: Simulation Study on the Benefits of Intention Sharing*. [Online]. Available: https://open.rijkswaterstaat.nl/overige-publicaties/2022/digital-intention-sharing-simulation/

[9] STM.(2019). *Test Report: Ship to Ship Route Exchange (S2SREX)-Controlled Simulation Trials, STM_ID3.3.8. Sea Traffic Management (STM)*. [Online]. Available: https://stm-stmvalidation.s3.eu-west-1.amazonaws.com/uploads/20190403113935/STM_ID3.3.8-Test-Report_Ship-to-Ship-Route-Exchange_ver_2.pdf

[10] M. Akdağ, T. I. Fossen, and T. A. Johansen, "Collaborative collision avoidance for autonomous ships using informed scenario-based model predictive control," *IFAC-PapersOnLine*, vol. 55, no. 31, pp. 249–256, 2022.

[11] H. Zheng, R. R. Negenborn, and G. Lodewijks, "Fast ADMM for distributed model predictive control of cooperative waterborne AGVs," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 4, pp. 1406–1413, Jul. 2017.

[12] Z. Du, R. R. Negenborn, and V. Reppa, "COLREGS-compliant collision avoidance for physically coupled multi-vessel systems with distributed MPC," *Ocean Eng.*, vol. 260, Sep. 2022, Art. no. 111917.

[13] L. Chen, R. R. Negenborn, and H. Hopman, "Intersection crossing of cooperative multi-vessel systems," *IFAC-PapersOnLine*, vol. 51, no. 9, pp. 379–385, 2018.

[14] M. Kurowski et al., "Multi-vehicle guidance, navigation and control towards autonomous ship maneuvering in confined waters," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 2559–2564.

[15] H. A. Tran, T. A. Johansen, and R. R. Negenborn, "Parallel distributed collision avoidance with intention consensus based on ADMM," *IFAC-PapersOnLine*, vol. 58, no. 10, pp. 302–309, 2024.

[16] L. Chen, H. Hopman, and R. R. Negenborn, "Distributed model predictive control for vessel train formations of cooperative multi-vessel systems," *Transp. Res. C, Emerg. Technol.*, vol. 92, pp. 101–118, Jul. 2018.

[17] B.-O.-H. Eriksen, G. Bitar, M. Breivik, and A. M. Lekkas, "Hybrid collision avoidance for ASVs compliant with COLREGs rules 8 and 13–17," *Frontiers Robot. AI*, vol. 7, Feb. 2020, Art. no. 11, doi: 10.3389/frobt.2020.00011.

[18] S. Yu, M. Hirche, Y. Huang, H. Chen, and F. Allgöwer, "Model predictive control for autonomous ground vehicles: A review," *Auto. Intell. Syst.*, vol. 1, no. 1, p. 4, Aug. 2021.

[19] L. Ferranti, L. Lyons, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, "Distributed nonlinear trajectory optimization for multi-robot motion planning," *IEEE Trans. Control Syst. Technol.*, vol. 31, no. 2, pp. 809–824, Mar. 2023.

[20] L. Ferranti, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, "Coordination of multiple vessels via distributed nonlinear model predictive control," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2018, pp. 2523–2528.

[21] S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.

[22] A. Themelis and P. Patrinos, "Douglas-rachford splitting and ADMM for nonconvex optimization: Tight convergence results," *SIAM J. Optim.*, vol. 30, no. 1, pp. 149–181, Jan. 2020.

[23] H. A. Tran, T. A. Johansen, and R. R. Negenborn, "Distributed MPC for autonomous ships on inland waterways with collaborative collision avoidance," 2024, *arXiv:2403.00554*.

[24] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Trans. Signal Process.*, vol. 62, no. 7, pp. 1750–1761, Apr. 2014.

[25] A. Makhdoumi and A. Ozdaglar, "Convergence rate of distributed ADMM over networks," *IEEE Trans. Autom. Control*, vol. 62, no. 10, pp. 5082–5095, Oct. 2017.

[26] N. Bastianello, R. Carli, L. Schenato, and M. Todescato, "Asynchronous distributed optimization over lossy networks via relaxed ADMM: Stability and linear convergence," *IEEE Trans. Autom. Control*, vol. 66, no. 6, pp. 2620–2635, Jun. 2021.

[27] T.-H. Chang, M. Hong, W.-C. Liao, and X. Wang, "Asynchronous distributed ADMM for large-scale optimization—Part I: Algorithm and convergence analysis," *IEEE Trans. Signal Process.*, vol. 64, no. 12, pp. 3118–3130, Jun. 2016.

[28] M. Hong, "A distributed, asynchronous, and incremental algorithm for nonconvex optimization: An ADMM approach," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 3, pp. 935–945, Sep. 2018.

[29] H. Wang, S. Marella, and J. Anderson, "FedADMM: A federated primal-dual algorithm allowing partial participation," in *Proc. IEEE 61st Conf. Decision Control (CDC)*, Dec. 2022, pp. 287–294.

[30] Q. Tran-Dinh, N. H. Pham, D. T. Phan, and L. M. Nguyen, "FedDR-randomized Douglas–Rachford splitting algorithms for nonconvex federated composite optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2021, pp. 30326–30338.

[31] H. A. Tran, T. A. Johansen, and R. R. Negenborn, "A collision avoidance algorithm with intention prediction for inland waterways ships," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 4337–4343, 2023.

[32] BPR.(2017). *Binnenvaartpolitiereglement*. [Online]. Available: https://wetten.overheid.nl/BWBR0003628/2017-01-01

[33] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, Mar. 2019.

[34] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar. 2006.

[35] N. Lauvås, H. A. Urke, and J. A. Alfredsen, "Design and validation of a system of autonomous fish tracking vehicles," in *Proc. Oceans*, Oct. 2022, pp. 1–10.

[36] J. Pinto, P. S. Dias, R. Martins, J. Fortuna, E. Marques, and J. Sousa, "The LSTS toolchain for networked vehicle systems," in *Proc. MTS/IEEE OCEANS Bergen*, Jun. 2013, pp. 1–9.

[37] M. Akdağ, H. A. Tran, N. Lauvås, T. A. Pedersen, T. I. Fossen, and T. A. Johansen, "A decentralized negotiation protocol for collaborative collision avoidance of autonomous surface vehicles," *TechRxiv*, May 2024, doi: 10.36227/techrxiv.171504685.50505720/v1.