


Decentralized Multi-Robot Formation Control with Communication Delay and Asynchronous Clock

Long Peng  · Fei Guan · Luc Perneel · Hasan Fayyad-Kazan · Martin Timmerman

Received: 29 July 2016 / Accepted: 5 April 2017 / Published online: 24 April 2017
© Springer Science+Business Media Dordrecht 2017

Abstract This paper investigates the leader–follower formation control problem for a group of networked nonholonomic mobile robots that are subject to bounded time-varying communication delays and an asynchronous clock. First we convert the formation control problem into a trajectory tracking problem, and then a fully distributed unified control framework based on the receding horizon control is implemented to converge the tracking errors. By adding an auxiliary acceleration term into the receding horizon controller, the framework is able to solve the impractical velocity jump problem. Considering the time-varying delays, the timing and order features of the messages are utilized to guarantee their logical correctness. To compensate for the delay effect, an improved control framework that exploits the predictability of the receding horizon controller is proposed. The asynchronous clock problem, which makes the communication delay unmeasurable, is studied. We give a definition of the syn point that is inspired from investigation of the property that messages are received out of order in

a bounded time-varying delayed network. A novel method that detects the occurrence of syn points is integrated into the control framework to solve the asynchronous clock problem. Finally the effectiveness of the proposed approaches is demonstrated in the Player/Stage simulation environment.

Keywords Leader–follower formation · Trajectory tracking · Receding horizon control · Communication delay · Asynchronous clock · Syn point

1 Introduction

A multi-robot system, with its intrinsically distributed and concurrent characteristics, is deemed to be more robust and efficient than a single versatile robot when performing some cooperative tasks such as search and rescue, surveillance, construction, and sampling [2, 17, 26, 35, 39]. An interesting and significant issue in this realm is to drive multiple autonomous robots to form, and move in, a specific geometric shape: this is referred to as formation control. Over the past two decades, a great number of studies have been devoted to this problem, mainly focusing on three approaches: the behavior-based approach [1, 40], the virtual structure approach [22, 30, 43], and the leader–follower approach [7, 16, 28, 32, 41]. Due to its simplicity and scalability, the leader–follower approach is widely adopted and has become predominant. Within this approach, the interaction between a group of n robots can be modeled

L. Peng (✉) · F. Guan · L. Perneel · H. Fayyad-Kazan
Department of Electronics and Informatics,
Vrije Universiteit Brussel (VUB), Pleinlaan 2,
Brussel 1050, Belgium
e-mail: Long.Peng@vub.ac.be

L. Peng
National Key Laboratory of Parallel and Distributed
Processing, National University of Defense Technology,
Changsha 410073, China

as a tree structure. One robot is designated as the root node, which determines the trajectory of the whole group. Each of the other robots, referred to as a “child” node in the tree, will maintain the desired separation and relative bearing angle from its “father” node. In all, there are $n - 1$ father–child pairs, i.e., leader–follower relationships. As a result, the formation can be achieved by locally coordinating each follower’s velocity according to its own state and the leader’s state. When implementing the leader–follower approach, there are generally two categories with respect to the control architecture. They are referred to as centralized and decentralized architectures. In a centralized architecture, a central monitoring node is assigned to acquire the state information of all the group members, compute the commands, and communicate with each robot. In a decentralized architecture, each robot calculates control commands by employing its own processing capacity according to its leader’s state, and moves autonomously without resorting to the central node. The decentralized solution outperforms the centralized one in terms of reliability and scalability [29, 34, 41]. The central monitoring node impacts on the reliability of the whole system as it is a single point of failure, and impacts on the scalability of the system as the number of robots is limited by the communication bandwidth and processing power of this central node. Therefore, this paper focuses on the decentralized leader–follower approach.

Following the leader–follower pattern, the formation control problem is naturally an extension of a traditional trajectory tracking problem, where the waypoints of the trajectory of the follower are calculated based on the state of its leader. Various novel tracking control strategies have been proposed to guarantee the stability of the system, such as I/O linearization [9, 41], backstepping [23, 28], sliding mode [10, 31], and receding horizon [4, 7, 12]. To achieve decentralized formation control, one fundamental requirement must be met: every follower must be aware of information about its leader’s state. To obtain such information, one general approach is to measure the leader’s state with the follower’s onboard vision sensor [9, 16, 32, 38]. However, due to the rapid development of wireless communication technologies, it is desirable to endow the autonomous robots with the capability of communicating. Thus, the leader is able to measure its state information by using its local sensors and send it to the followers. In real implementations, a problem

that can not be ignored is the delay in the exchange of information between the robots. Without deliberately taking this into account, the communication delay will not only degenerate the performance of the system, but also affect its stability when the delay is large enough [21, 33, 41]. For instance, in the I/O linearization and backstepping techniques, the velocity control commands are directly related to the tracking errors. Large initial velocities can be generated when there are large initial tracking errors, and a sharp velocity jump could happen when sudden tracking errors arise. As a result, the required acceleration could exceed the physical specification of the robot, which makes the robot suffer from the impractical velocity jump problem [28, 36]. The communication delay could make it even worse as it further augments the tracking errors. Especially when the communication delay is varying over time, impractical velocity jumps could happen frequently, even though there are no tracking errors.

The effect of communication delays on the stability problem has been extensively studied in multi-agent systems [11, 25, 27, 37]. Jiang et al. [21], following the approach in [25], exploits a predictive term in the control law to get more accurate position information provided that every robot is fully aware of the fixed reference velocity. Izadi et al. [19, 20] proposes a fault-tolerant receding horizon controller to cope with the communication delays of predicted states of the leader robot. It requires transmitting the predicted states over the whole prediction horizon, and also estimating the tail part of the leader robot’s state that is unavailable due to communication delays. Concerning the leader–follower formation control of nonholonomic robots, Xu et al. [41] proposes a PD-type controller based on previous states to compensate for the delayed effect, which conforms to the work in [11, 27]. However, to the best of our knowledge, there is nothing in the literature that treats the asynchronous clock problem for formation control in a distributed environment. In a multi-robot system that uses embedded processors, the robots are likely to have different internal clocks and timing mechanisms, which makes the communication delay difficult to measure.

This paper studies the leader–follower formation control problem of a group of networked nonholonomic mobile robots subject to bounded time-varying communication delay and asynchronous clocks. The main contributions of this paper can be summarized as follows.

- Based on the receding horizon control principle and ACADO tool [18], we implement a fully distributed unified control framework to address the leader–follower formation control problem. By adding an auxiliary acceleration term into the receding horizon controller, the framework solves the impractical velocity jump problem.
- To solve the bounded time-varying communication delay problem, an improved framework that exploits the predictability of the receding horizon control is proposed. As the delay is time-varying, the follower will receive messages out of order. The timing and order features are utilized to guarantee the correct temporal order of messages on the follower side.
- The feature that at a sampling time the follower may receive several messages in a bounded time-varying communication network is analysed. We give the definition of a syn point. By detecting the occurrence of syn points, the framework solves the formation control problem regardless of having asynchronous clocks among the robots. In other words, there is no need to strictly synchronize the clocks by using a delicate clock synchronization protocol for a fully distributed networked robotic system that are subject to bounded time-varying communication delay and asynchronous clocks.

The rest of the paper is organized as follows. In Section 2 we describe the kinematic model of the nonholonomic robot, and formulate the leader–follower formation control problem. The general form of the receding horizon control is presented in Section 3, and then the unified control framework integrating a receding horizon control scheme is presented, as well as the algorithms proposed to solve the communication delay and the asynchronous clock problems. The simulation results are reported in Section 4, followed by some concluding remarks in Section 5.

2 Leader–follower Formation Control

2.1 Mathematical model of the mobile robot

Throughout this paper, we consider a multi-robot system with N mobile robots. Every R_i ($1 \leq i \leq N$), as

shown in Fig. 1, is a differential driven robot. The posture state p_i of R_i is fully described by a 3-D column vector

$$p_i = [x_i, y_i, \theta_i]^T, \tag{1}$$

where x_i and y_i denotes the coordinates of the midpoint between the two wheels with respect to an inertial coordination frame, and θ_i is the orientation angle.

We assume that each robot moves under pure rolling and non-slipping conditions [8], whereby the robot R_i has the following nonholonomic kinematic constraint:

$$\dot{x}_i \sin \theta_i - \dot{y}_i \cos \theta_i = 0. \tag{2}$$

Therefore, the kinematic model of R_i can be described by

$$\dot{p}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \cos \theta_i & 0 \\ \sin \theta_i & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix}, \tag{3}$$

where v_i and ω_i are the linear velocity and the angular velocity, respectively.

2.2 The leader–follower control objective

In the leader–follower formation scheme, the follower in each leader–follower pair is required to maintain the desired separation and relative bearing angle with respect to the leader. To be more specific, consider the leader–follower pair in Fig. 2. The desired separation, respectively, bearing angle, between the follower

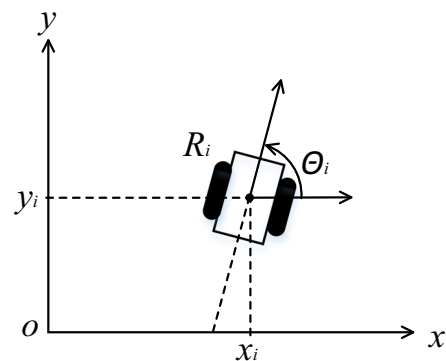


Fig. 1 The model of the nonholonomic robot

R_i and the leader R_j are denoted by ρ_d , respectively, φ_d , while ρ , respectively, φ , are the actual values. As a result, the control objective is to control the input $[v_i, \omega_i]^T$ of R_i in such a way that

$$\begin{cases} \rho - \rho_d \rightarrow 0 \\ \varphi - \varphi_d \rightarrow 0 \end{cases} \quad (4)$$

as $t \rightarrow \infty$.

From Fig. 2, it is obvious that ρ_d and φ_d uniquely determine the desired position of R_i . Besides, it is also desired that R_i maintain the same orientation angle as R_j . Therefore, the desired posture p_d of R_i can be calculated by

$$p_d = \begin{bmatrix} x_d \\ y_d \\ \theta_d \end{bmatrix} = \begin{bmatrix} x_j - \rho_d \cos(\varphi_d + \theta_j) \\ y_j - \rho_d \sin(\varphi_d + \theta_j) \\ \theta_j \end{bmatrix}. \quad (5)$$

The control objective can also be described as controlling $[v_i, \omega_i]^T$ to make p_i ($[x_i, y_i, \theta_i]^T$) track the desired reference trajectory determined by the kinematics of the point p_d ($[x_d, y_d, \theta_d]^T$).

In order to exploit feedback linearization, it is conventional to consider the kinematics of a handling point p'_i that lies on the orientation axis at a distance

d from p_i [9, 32]. In this way, the kinematics of a robot's coordinates is explicitly related to its angular velocity. Following this convention, the control objective is changed into controlling $[v_i, \omega_i]^T$ to make p'_i track the desired reference trajectory of point p'_d , where

$$p'_i = \begin{bmatrix} x'_i \\ y'_i \\ \theta'_i \end{bmatrix} = \begin{bmatrix} x_i + d \cos \theta_i \\ y_i + d \sin \theta_i \\ \theta_i \end{bmatrix} \quad (6)$$

and

$$p'_d = \begin{bmatrix} x'_d \\ y'_d \\ \theta'_d \end{bmatrix} = \begin{bmatrix} x_j - \rho_d \cos(\varphi_d + \theta_j) + d \cos \theta_j \\ y_j - \rho_d \sin(\varphi_d + \theta_j) + d \sin \theta_j \\ \theta_j \end{bmatrix}. \quad (7)$$

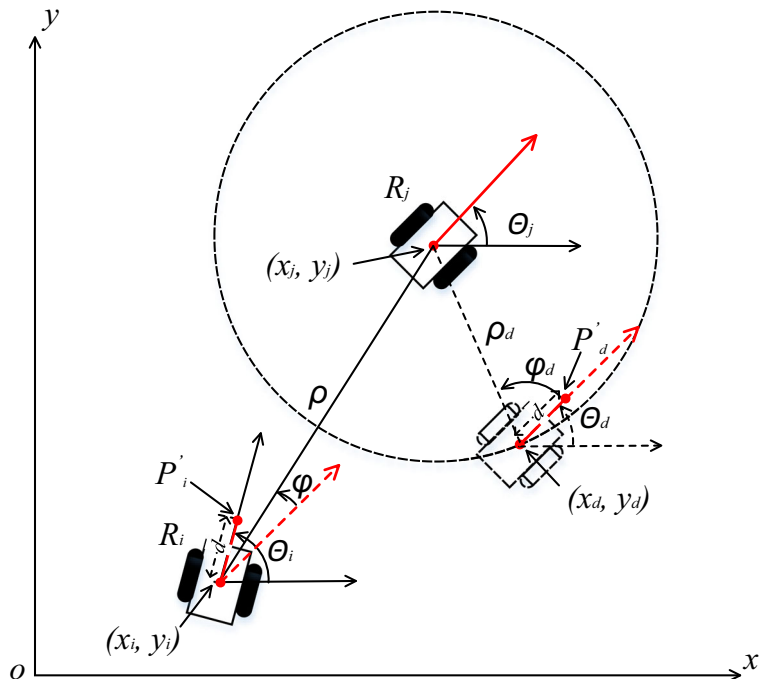
Thus, the tracking error e can be defined as

$$e = \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} = A(p'_d - p'_i) \quad (8)$$

where

$$A = \begin{bmatrix} \cos \theta_i & \sin \theta_i & 0 \\ -\sin \theta_i & \cos \theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (9)$$

Fig. 2 A leader–follower pair



As A is invertible, it is obvious that $e \rightarrow \mathbf{0}$ implies $(p'_d - p'_i) \rightarrow \mathbf{0}$. Combining (6), (7), (8), and (9), the tracking error can be rewritten as

$$\begin{aligned} e_x &= (x_j - x_i) \cos \theta_i + (y_j - y_i) \sin \theta_i \\ &\quad - \rho_d \cos(\varphi_d + \theta_j - \theta_i) + d \cos(\theta_j - \theta_i) - d, \\ e_y &= (y_j - y_i) \cos \theta_i - (x_j - x_i) \sin \theta_i \\ &\quad - \rho_d \sin(\varphi_d + \theta_j - \theta_i) + d \sin(\theta_j - \theta_i), \\ e_\theta &= \theta_j - \theta_i. \end{aligned} \tag{10}$$

By taking the derivative of Eq. 10 and using Eq. 2, the dynamics of the tracking error can be obtained as

$$\begin{aligned} \dot{e}_x &= v_j \cos e_\theta + e_y \omega_i - v_i \\ &\quad + \rho_d \omega_j \sin(\varphi_d + e_\theta) - d \omega_j \sin e_\theta, \\ \dot{e}_y &= v_j \sin e_\theta - e_x \omega_i - d \omega_i \\ &\quad - \rho_d \omega_j \cos(\varphi_d + e_\theta) + d \omega_j \cos e_\theta, \\ \dot{e}_\theta &= \omega_j - \omega_i. \end{aligned} \tag{11}$$

We rewrite the dynamics of the tracking error as a general nonlinear system

$$\dot{e}(t) = f_e(e(t), u_i(t)), \quad t \geq 0, \tag{12}$$

where the input vector $u_i(t) = [v_i, \omega_i]^T$.

3 Controller Design

3.1 Receding horizon control principle

Receding horizon control, also referred to as model predictive control (MPC), aims to solve a finite horizon open-loop optimal control problem that is subject to system dynamics and input and state constraints [3]. Figure 3 shows the general principle of receding horizon control. At time t_k ($t_k = t_0 + k\delta$, where t_0 is the initial time, δ is the sampling period, and $k \in \mathbb{N}$), according to the current state of the system, the controller predicts the future state trajectory over a prediction horizon (T_p), and also generates the future input trajectory over a control horizon ($T_c \leq T_p$; for simplicity, we assume $T_c = T_p$ in this paper) with the purpose of minimizing a cost function J , while the evolution of the system is subject to the system's

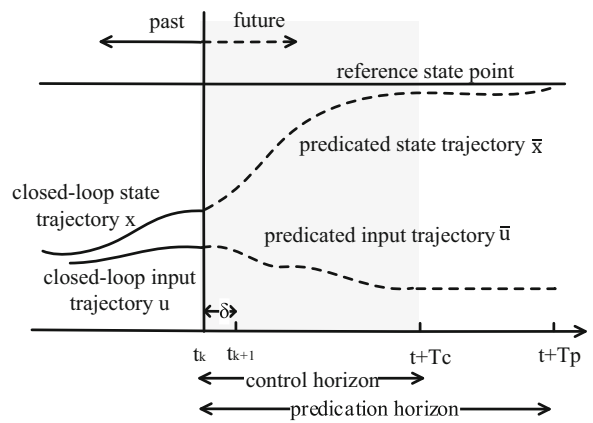


Fig. 3 General receding horizon principle

dynamic model and state and input constraints. Due to the idealization of the model and the uncertainty of the disturbances, in most cases the real state trajectory will not follow the predicted one. As a result, the generated input is only applied to the system over the sampling period δ , after which the receding horizon control algorithm is executed again with the new system state.

To be more specific, consider a general nonlinear system described by

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0, \tag{13}$$

and subject to the state and input constraints

$$x(t) \in X, \quad \forall t \geq 0 \tag{14}$$

$$u(t) \in U, \quad \forall t \geq 0, \tag{15}$$

where $X \subset \mathbb{R}^n$ and $U \subset \mathbb{R}^m$. At the sampling time t_k , the receding horizon control solves a finite horizon open-loop optimal control problem, formulated as

$$\min_{\bar{u}(\cdot)} J(x(t_k), \bar{u}(\cdot)) \tag{16a}$$

with

$$\begin{aligned} J(x(t_k), \bar{u}(\cdot)) &= \int_{t_k}^{t_k+T_p} F(\bar{x}(\tau), \bar{u}(\tau)) d\tau \\ &\quad + E(\bar{x}(t_k + T_p)) \end{aligned} \tag{16b}$$

subject to

$$\dot{\bar{x}}(\tau) = f(\bar{x}(\tau), \bar{u}(\tau)), \quad \bar{x}(t_k) = x(t_k) \quad (16c)$$

$$\bar{x}(\tau) \in X, \quad \bar{u}(\tau) \in U, \quad \tau \in [t_k, t_k + T_p] \quad (16d)$$

$$\bar{x}(t_k + T_p) \in \Omega, \quad \Omega \subseteq X, \quad (16e)$$

where $\bar{x}(\cdot)$ is the predicted state trajectory over the prediction horizon driven by $\bar{u}(\cdot) : [t_k, t_k + T_p] \rightarrow U$; $F : X \times U \rightarrow \mathbb{R}^+$ is called the stage cost function: it is assumed to be continuous. Usually F is expressed as a quadratic form:

$$F(\bar{x}(\tau), \bar{u}(\tau)) = \bar{x}(\tau)^T Q \bar{x}(\tau) + \bar{u}(\tau)^T R \bar{u}(\tau), \quad (17)$$

where $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ are positive-definite and positive-semidefinite weighting matrices, respectively; E and Ω denote the terminal penalty function and terminal region, respectively. These two parameters need to be carefully chosen in order to guarantee close-loop stability. Theorem 1 below states the stability conditions.

Theorem 1 ([13]) *Consider the design parameters: the prediction time horizon T_p , the stage cost function F , the terminal penalty term E , and the terminal region Ω , which satisfy*

1. $\Omega \subseteq X$ is closed with $\mathbf{0} \in \Omega$. E is positive semi-definite;
2. $\forall x(0) \in \Omega$ there exists an input $u_\Omega : [0, \delta] \rightarrow U$ such that

$$x(\tau) \in \Omega, \quad \forall \tau \in [0, \delta] \quad (18)$$

and

$$\frac{\partial E}{\partial x} f(x(\tau), u_\Omega(\tau)) + F(x(\tau), u_\Omega(\tau)) \leq 0, \quad \forall \tau \in [0, \delta]; \quad (19)$$

3. The finite horizon open-loop optimal problem (16a) is feasible (has a solution) at $t = 0$.

Then Eq. 16a is feasible for all $t \geq 0$ and $x(t)$ will converge to zero as $t \rightarrow \infty$ when repeatedly applying the optimal solution produced by solving (16a) at each sampling time $t_k, k \in \mathbb{N}$.

Proof See [13]. □

3.2 Delay-free receding horizon leader–follower control framework

With an ideal communication channel, the communication delay between the leader and follower can be ignored. We also assume that the robot is equipped with sensors making state (posture and velocity) measurement feasible. In this situation, we propose a distributed control framework without considering communication delay as shown in Fig. 4. The low level controller is in charge of adjusting the robot’s velocity assigned by the high level controller, as well as measuring its state according to the sensor data. Taking only the kinematics of the robot into account, the high level controller employs the receding horizon controller to calculate its velocity according to the state information at each sampling step.

The leader robot moves at a reference velocity (v_r, ω_r) , while the follower is required to converge to and then maintain a desired distance and bearing angle relative to the leader. It is obvious that they have different control objectives, which are deemed to be difficult to implement by using the same receding horizon controller. However, the proposed approach in Fig. 4 is a unified one that can be applied to both the leader and the follower. To be more specific, all the robots have the same receding horizon approach that accepts unified parameters. The high level logic supplies the receding horizon controller with different values adaptively according to the specific control objective. The overall method is detailed in Algorithm 1. The core idea is to imagine a virtual leader that always has the same position as the leader robot, but moves in terms of (v_r, ω_r) . The leader robot’s control objective is to track the virtual leader with zero separation and zero bearing angle. In this way, we are able to implement the same receding horizon controller in both the leader and the follower. Despite its high computational and memory cost, a receding horizon controller outperforms a generic PID controller in terms of state and input constraints and the prediction capability. Furthermore, its prediction capability makes it natively capable of handling the communication delay problem as detailed in Section 3.3.

Algorithm 1 The unified approach integrating the receding horizon controller

Data: The reference velocity (v_r, ω_r) ; the desired separation ρ_d and bearing angle φ_d .

Result: Stable formation (ρ_d, φ_d) between the leader and the follower.

```

1 if leader then
2   |  $\rho_d \leftarrow 0; \varphi_d \leftarrow 0;$ 
3 else
4   |  $\rho_d \leftarrow \rho_d; \varphi_d \leftarrow \varphi_d;$ 
5 while !stop do
6   | At sampling instant  $t_k$ , get the robot's state
   | from the low level controller;
7   | if leader then
8     | send a message  $msg_k$  that includes its
8     | state information to the follower;
9     | set a virtual leader for the leader robot.
   | The virtual leader has the same position
   |  $(x_j, y_j, \theta_j)$  as the leader, but is assigned
   | the reference velocity  $(v_r, \omega_r)$ ;
10    | calculate the tracking error  $(e_x, e_y, e_\theta)$ 
   | between the leader and the virtual leader
   | according to (10);
11  | else
12    | receive the leader's message  $msg_k$ ;
13    | calculate the tracking error  $(e_x, e_y, e_\theta)$ 
   | between the leader and the follower
   | according to (10);
14  | calculate the input  $\bar{u}(t) : (t_k \leq t \leq t_k + T_p)$ 
   | by applying (12) and (16a);
15  | send the input  $\bar{u}(t) : (t_k \leq t \leq t_k + \delta)$  only
   | within the sampling period to the low level
   | controller;
16  |  $k \leftarrow (k + 1);$ 

```

Regarding the implementation, we use the ACADO toolkit [18] to generate portable, self-contained C code for the receding horizon controller as shown in Fig. 5. To solve the impractical velocity jump problem [28], we change the error dynamics by introducing auxiliary acceleration terms. Accordingly, v_i and ω_i are treated as state variables, and the accelerations are considered as the inputs (lines 3 and 5 in Fig. 5). In this way, the changing rate of velocity is constrained

by the acceleration value (line 12), which naturally prevents the velocity from jumping too sharply. Additionally, Lines 29 and 30 express the acceleration and velocity constraints. We also regard ρ_d and φ_d as variables instead of constants (line 4), making the system adaptable to diverse formation objectives at run-time. For the cost function J , we only take the error states and accelerations into account (lines 17–21). Such a configuration makes the tracking error dominant in the cost function, since the main objective is to minimize the tracking error. Meanwhile, it also makes the acceleration converge to zero. In other words, it will keep the velocity as stable as possible while minimizing the tracking error.

3.3 Improved control framework subject to communication delay

The control framework proposed in Section 3.2 assumes that the follower can receive the leader's state messages without delay and loss. However, this assumption is not sustainable in most real systems, wherein the system inevitably suffers from constant or time-varying delays due to an imperfect communication network. Without taking the communication delay into account, it is very possible that the tracking error will not converge to zero, because the follower will always track a delayed leader. More specifically, when the delays are time-varying, the follower will receive the state messages in a random order, which results in instability (see the simulation result in Section 4).

In this paper, we deal with a multi-robot system subject to time-varying but bounded communication delay. First, we follow the approach in [24] to model the communication delay problem from the the follower's perspective. For simplicity, we suppose that msg_k is the latest message the leader has sent at the sampling instant t_k for the follower, and use $msg_k.stime$ to denote its sending time. We also assume that the leader and the follower have the same logical clock. As Algorithm 1 describes, at time t_k , the follower is supposed to receive msg_k from the leader. However, due to the time-varying delay, msg_k is likely to be received after t_k , whereas at time t_k the follower may receive several messages prior to msg_k , or

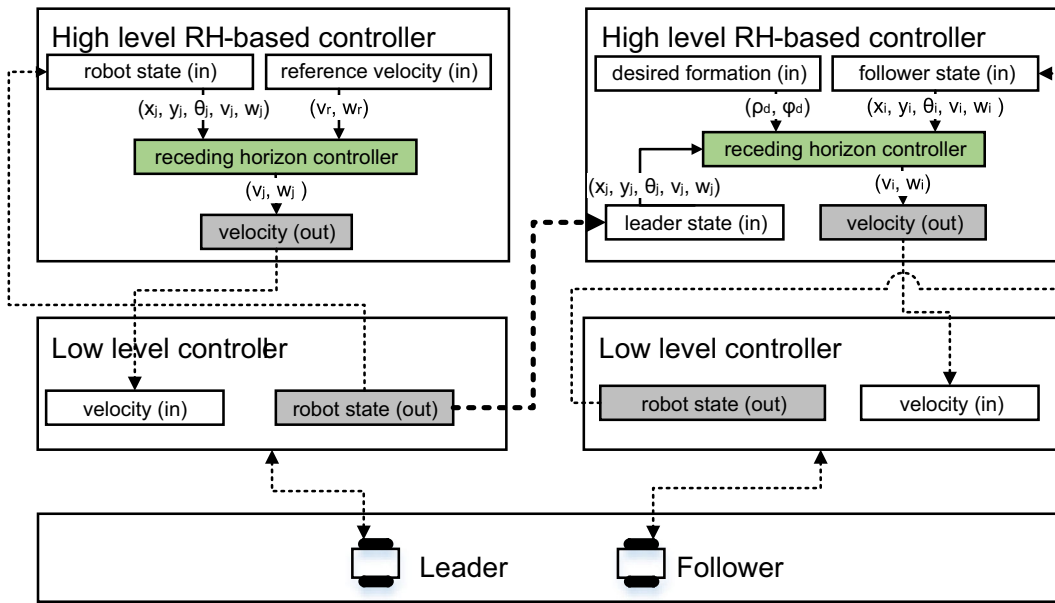


Fig. 4 The unified distributed leader–follower control framework without considering communication delay

even receive nothing. For a message msg_i received at time t_k , we use an auxiliary variable d_i , the number of sampling periods, to indicate its delay. As the delay is bounded, we suppose that

$$\underline{D} \leq d_i \leq \overline{D}, \underline{D} \in \mathbb{N}, \overline{D} \in \mathbb{N}, \tag{20}$$

where \underline{D} and \overline{D} are the minimum and maximum delays, respectively. Since the receding horizon controller is implemented discretely, d_i is calculated approximately by

$$d_i = \lfloor \frac{t_k - msg_i.stime + 0.5 * \delta}{\delta} \rfloor. \tag{21}$$

In order to compensate for the delay effect, in general there are two kinds of approaches: introducing a self-delay [5, 11, 37, 41], and introducing a prediction term [21, 25] in the control laws. In this paper we employ a prediction approach that seamlessly integrates with the receding horizon controller due to its inherent prediction capability. The state and input trajectories, as a byproduct of the receding horizon control, can provide us with a reliable future state of the leader robot. In principle, the prediction can be embedded into the control framework either on the leader’s side or on the follower’s side. If implemented on the follower’s side, the control architecture of the leader needs no modification, and the follower will receive a delayed state from the leader.

Therefore, the receding horizon controller of the follower should consider the delay of messages. Not too much effort is needed to achieve this. However, it is simplistic and incomplete in this case without considering unforeseen situations on the leader side. For instance, if the leader encounters an obstacle, its future state will possibly deviate from that estimated by the follower, whereas the follower is unaware of the situation, which consequently results in a deviation for the formation. In contrast, if implemented on the leader side, the obstacles can be detected and avoided using an on-line receding horizon strategy [14, 42], which, of course, provides more accurate future states. We prefer to implement the prediction strategy on the leader’s side, although avoiding obstacles is not the focus of this paper.

Figure 6 shows the improved distributed control framework with the prediction capability embedded. In order to get the robot’s predicted state trajectory, we modify the receding horizon controller just by adding the dynamics of the robot to the ACADO code. After the trajectory is calculated, the next step is to determine which element of the predicted states should be sent to the follower. However, this is a dilemma for the leader, because it is incapable of knowing the actual delay in advance. We solve this problem by sending the predicted state that is exactly beyond \overline{D} , the upper bound of the communication

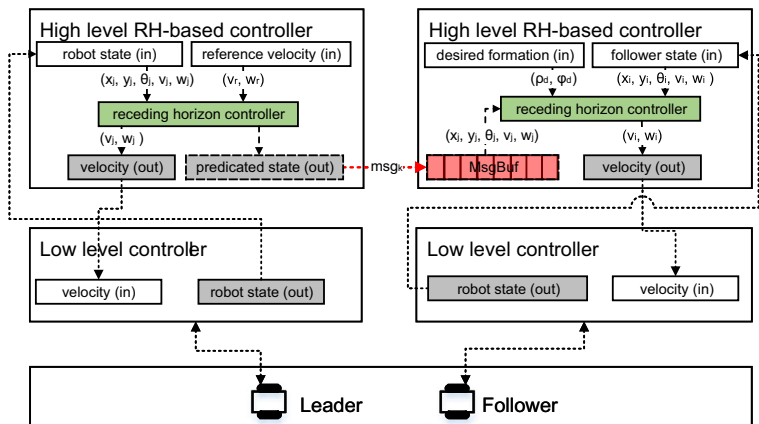

```

/* Variables */
DifferentialState e_x, e_y, e_theta; // the tracking error
DifferentialState v_i, w_i, v_j, w_j; // linear and angular
    velocities
DifferentialState rho_d, alpha_d, d; // desired parameters.
Control av_i, aw_i; // the acc.
of the linear and angular velocity
/* Model equations */
DifferentialEquation f;
f << dot(e_x ) = -v_i + e_y*w_i + v_j*cos(e_theta) + rho_d*w_j*
    sin(alpha_d+e_theta)-d*w_j*sin(e_theta);
f << dot(e_y ) = (-e_x-d)*w_i + v_j*sin(e_theta) - rho_d*w_j*cos(
    alpha_d+e_theta)+d*w_j*cos(e_theta);
f << dot(e_theta ) = w_l - w_f;
f << dot(v_i ) = av_i; f << dot( w_i ) = aw_i;
f << dot(v_j ) = 0; f << dot(w_j) = 0;
f << dot(rho_d) = 0; f << dot(alpha_d) = 0; f << dot(d) = 0;
/* Reference functions and weighting matrices */
Function h, hN;
h << e_x << e_y << e_theta << av_i << aw_i;
hN << e_x << e_y << e_theta;
DMatrix W = eye<double>( h.getDim() );
/* 0.5, 0.5, 0.5, 0.2, 0.2 */
W(0,0) = Q_X; W(1,1) = Q_Y; W(2,2) = Q_THETA; W(0,0) = R_V; W(0,0)
    = R_W;
DMatrix WN = eye<double>(hN.getDim()); WN *= 1;
/* Optimal Control Problem */
OCP ocp(0.0, 1.0, 10); //one second prediction
ocp.subjectTo(f);
ocp.minimizeLSQ(W, h); ocp.minimizeLSQEndTerm(WN, hN);

ocp.subjectTo(AV_MIN <= av_i <= AV_MAX); ocp.subjectTo(AW_MIN <=
    aw_i <= AW_MAX);
ocp.subjectTo(V_MIN <= v_i <= V_MAX); ocp.subjectTo(W_MIN <= w_i
    <= W_MAX);
OCPexport mpc(ocp); mpc.exportCode();
    
```

Fig. 5 The ACADO code to generate the source code for the leader–follower control problem

Fig. 6 The improved distributed leader–follower control framework embedded with prediction capability to compensate for the effects of communication delay



delay. With this strategy, in most cases the data received at time t_k by the follower is not the immediate desired state data, thus a loop message buffer $MsgBuf$ (see Fig. 6) is employed to cache the data, the size of which depends on \overline{D} . At each sampling time, the follower will get the immediate predicted state of the leader from the buffer.

Due to the randomness of communication delay, there are two issues that must be addressed: 1) sorting the messages received in the order of their sending time in the buffer; 2) putting a message into its correct position in $MsgBuf$. We denote the position for a message msg_i in the buffer by pos_i . At first sight, the two issues can be easily solved by utilizing the time-stamps of the messages. Suppose that at the sampling time t_k , an index variable idx , which moves at the sampling rate, points to the right message that contains the leader’s state information in $MsgBuf$. When receiving a message msg_i , pos_i can be calculated by

$$pos_i = (idx + \overline{D} - d_i) \bmod (\overline{D} + 1), \tag{22}$$

where d_i is the communication delay and is obtained from Eq. 21. The correctness of Eq. 22 is, inevitably, decided by the accuracy of Eq. 21. If the leader sends messages strictly according to the sampling rate, namely, for two sequential messages msg_i and msg_{i+1} we have

$$msg_{i+1}.stime = msg_i.stime + \delta, \tag{23}$$

then, after some calculations, we have

$$pos_{i+1} = (pos_i + 1) \bmod (\overline{D} + 1). \tag{24}$$

In this manner all the messages are correctly sorted and settled in $MsgBuf$. However, Eq. 24 will not be valid when there are some little jitters with the sending time. Figure 7 illustrates the scenario.

The two sequential messages are received at sampling time t_{k+1} and t_k , respectively. The little jitters

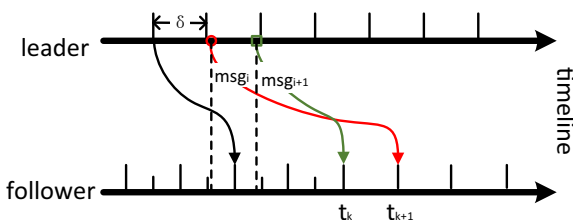


Fig. 7 An example that generates the same position for two sequential messages. msg_i is sent a little late, while msg_{i+1} is sent a little early

will definitely make their positions identical by simply applying (22). Actually the analysis above only relies on the timing feature of the messages, without taking their sending order into account. To solve this problem, we endow the messages with an additional auxiliary field seq that indicates their sending order. For any two sequential messages msg_i and msg_{i+1} sent by the leader, they always have

$$msg_{i+1}.seq = msg_i.seq + 1. \tag{25}$$

Algorithm 2 Using the timing and order features of messages to guarantee the logical correctness of storing messages in $MsgBuf$ on the follower’s side

Data: $MsgBuf[\overline{D} + 1] = \{0\}; idx = 0;$
 $seq = 0; pos = 0; k = 0.$

Result: the current state predicted at $MsgBuf[idx]$.

```

1 while !stop do
2   At sampling instant  $t_k$ , receive messages sent
   from the leader;
3   for each message  $msg_i$  received do
4     if  $msg_i$  is the first one received then
5       calculate  $pos_i$  by (22);
6        $pos \leftarrow pos_i;$ 
7        $seq \leftarrow msg_i.seq;$ 
8        $MsgBuf[pos_i] \leftarrow msg_i;$ 
9     else
10       $pos_i \leftarrow$ 
11       $(pos + msg_i.seq - seq) \bmod (\overline{D} + 1);$ 
12       $MsgBuf[pos_i] \leftarrow msg_i;$ 
13      //  $MsgBuf[idx]$  has the
      desired leader state. The
      steps to calculate the
      input are omitted.
12   $k \leftarrow (k + 1);$ 
13   $idx \leftarrow (idx + 1) \bmod (\overline{D} + 1);$ 

```

Algorithm 2 shows the pseudo code that utilizes the timing and order features of messages to insert them into $MsgBuf$ correctly. The first message received is used to initialize pos and seq , and its position is calculated individually by Eq. 22. Subsequent messages are placed according to sequence number relative to that of the first message, which guarantees their timing order in $MsgBuf$.

3.4 Asynchronous clock problem

To the best of our knowledge, when studying the effect of communication delays on the formation control problem, nearly all the existing literature assumes either that the communication delay is constant between two robots (agents), or that it is measurable if time-varying. As stated before, this paper considers the case of bounded time-varying delay, which is more general in real situations. In a centralized control framework or in a simulated environment, all the controllers are implemented in the same machine. They can share the same clock to obtain the interval between any two time-stamped events, and therefore we can utilize (21) to measure the delay of messages. However, in a distributed environment, except when the two robots have synchronized clocks, it is no longer correct to calculate the delay by means of Eq. 21 directly. Their local internal clocks may differ, which is referred to as the asynchronous clock problem, making the the communication delay unmeasurable. In this case, even a clock drift of one second could make the formation unstable. To solve this problem, a bunch of synchronization protocols have been designed to synchronize the asynchronous clocks. In this paper, instead of embedding a specified synchronization protocol into the control framework, we propose a syn point based algorithm to solve the asynchronous clock problem. The algorithm does not rely on the timing feature of the messages, and thereby it can work without any synchronization protocol. The efficacy and efficiency of this algorithm depends on the order feature of the messages and the distribution of the communication delays. To clarify the algorithm, we first give the definition of the **syn point**.

Definition 1 (Syn point). For the follower robot, a sampling time t_k is said to be a syn point if at time t_k it receives at least two messages, and there are two messages msg_m and msg_l such that

$$msg_m.seq - msg_l.seq = \bar{D} - \underline{D}. \tag{26}$$

The example in Fig. 8 shows that the sampling time t_x is a syn point, while t_{x-2} is not a syn point, even though the follower receives three messages at time t_{x-2} . It also shows that d_l (the delay of msg_l) is \bar{D} , and that d_m is equal to \underline{D} .

Theorem 2 For two messages msg_m and msg_l received at time t_k , $d_l - d_m = \bar{D} - \underline{D}$ if and only if $d_l = \bar{D}$ and $d_m = \underline{D}$.

Proof First, if $d_l = \bar{D}$ and $d_m = \underline{D}$, it is clear that $d_l - d_m = \bar{D} - \underline{D}$.

Second, if $d_l - d_m = \bar{D} - \underline{D}$, then $d_l = \bar{D} + (d_m - \underline{D})$. According to Eq. 20, we have $d_m \geq \underline{D}$, so $d_l \geq \bar{D}$. As $d_l \leq \bar{D}$, we can only have $d_l = \bar{D}$, as well as $d_m = \underline{D}$. \square

Lemma 1 t_k is a syn point if and only if at time t_k the following hold: 1) the follower receives at least two messages; 2) one message received has the maximum delay \bar{D} ; 3) one message received has the minimum delay \underline{D} .

Proof According to Definition 1, if t_k is a syn point, there are two messages msg_m and msg_l that satisfy

$$\begin{aligned} msg_m.seq - msg_l.seq &= \bar{D} - \underline{D}. \\ \iff msg_m.stime - msg_l.stime &= \delta * (\bar{D} - \underline{D}). \\ \iff msg_m.stime &= \delta * (\bar{D} - \underline{D}) + msg_l.stime. \end{aligned} \tag{27}$$

It follows from Eq. 21 that

$$d_m = \lfloor \frac{t_k - msg_m.stime + 0.5 * \delta}{\delta} \rfloor. \tag{28}$$

Combining (27) and (28), we get

$$\begin{aligned} d_m &= \lfloor \frac{t_k - \delta * (\bar{D} - \underline{D}) - msg_l.stime + 0.5 * \delta}{\delta} \rfloor. \\ \iff d_m &= \lfloor \frac{t_k - msg_l.stime + 0.5 * \delta}{\delta} \rfloor - (\bar{D} - \underline{D}). \\ \iff d_m &= d_l - (\bar{D} - \underline{D}). \\ \iff d_l - d_m &= (\bar{D} - \underline{D}). \end{aligned} \tag{29}$$

Using Theorem 2 and Eq. 29, we get $d_l = \bar{D}$ and $d_m = \underline{D}$. \square

Lemma 1 gives the impression that we are able to get the delay of messages through detecting the occurrence of syn points regardless of any synchronization protocols. Furthermore, a syn point can be easily identified by utilizing the *seq* field of the messages according to its definition. This is the key idea of Algorithm 3, which integrates the syn point method into the control framework. When a syn point is identified at time t_k , it indicates that the follower receives a message msg_l with the maximum delay \bar{D} . The

leader’s state inside msg_l , which was predicted exactly beyond \bar{D} at time $(t_k - \bar{D})$, is what the follower expects at time t_k . Therefore, we can adjust the index idx to make it point to the position where msg_l is stored in $MsgBuf$.

Algorithm 3 Integrating the **syn point** method into the control framework to solve the asynchronous clock problem

```

Data:  $MsgBuf[3 * \bar{D} + 1] = \{0\}$ ;
          $idx = 0, seq = 0, pos = 0, k = 0$ ;
          $seq_{min} = INFINITY, seq_{max} = 0, idx_{new} = 0$ ;
Result: the current state predicted at  $MsgBuf[idx]$ .
1 while !stop do
2   At sampling instant  $t_k$ , receive messages sent from the leader;
3    $seq_{min} = INFINITY, seq_{max} = 0, idx_{new} = idx$ ;
   // Put messages into  $MsgBuf$  in their sending order.
4   for each message  $msg_i$  received do
5     if  $msg_i$  is the first one received then
       // Initialization.
6        $pos \leftarrow pos_i \leftarrow (2 * \bar{D} + idx)$ ;
7        $seq \leftarrow msg_i.seq$ ;
8        $MsgBuf[pos_i] \leftarrow msg_i$ ;
9     else
10       $pos_i \leftarrow (pos + msg_i.seq - seq) \bmod (3 * \bar{D} + 1)$ ;
11       $MsgBuf[pos_i] \leftarrow msg_i$ ;
12     if  $msg_i.seq < seq_{min}$  then
13        $seq_{min} \leftarrow msg_i.seq$ ;
14        $idx_{new} \leftarrow pos_i$ ;
15     if  $msg_i.seq > seq_{max}$  then
16        $seq_{max} \leftarrow msg_i.seq$ ;
   // Identify a syn point.
17   if  $seq_{max} - seq_{min} = \bar{D} - \underline{D}$  then
18      $idx \leftarrow idx_{new}$ ;
   //  $MsgBuf[idx]$  has the desired leader state. The steps to calculate the input are omitted.
19    $k \leftarrow (k + 1)$ ;
20    $idx \leftarrow (idx + 1) \bmod (3 * \bar{D} + 1)$ ;

```

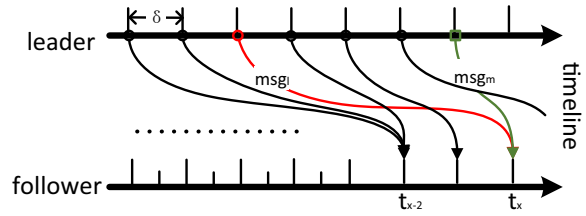


Fig. 8 An example that illustrates the occurrence of a syn point with $\delta = 100$ ms, the minimum delay = 100 ms ($\underline{D} = 1$), and the maximum delay = 500 ms ($\bar{D} = 5$)

Compared to embedding a specified synchronization protocol, the syn point based method is more lightweight. However, its efficiency depends on the distribution of the delays. Suppose that the delay variable d_i follows a discrete uniform distribution in $[\underline{D}, \bar{D}]$. Then the probability of the occurrence of a syn point at t_k is $1/(\bar{D} - \underline{D})^2$. In this situation, one syn point is deemed to appear within the time $\delta(\bar{D} - \underline{D})^2$. For instance, consider the configuration in Fig. 8. We can expect the occurrence of a syn point within 1.6 seconds.

4 Simulation Results

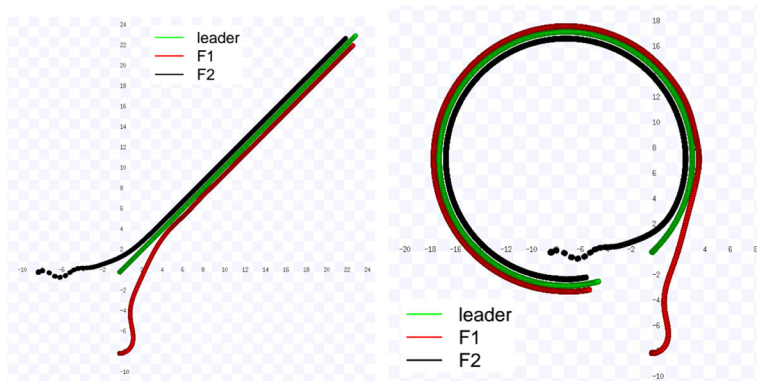
In this section, we present several simulations to validate the efficacy and effectiveness of the proposed three leader–follower formation control algorithms. The simulation experiments were carried out with several simulated Pioneer 2-DX robots using the Player/Stage software tools [6, 15]. The Stage simulation environment enables the robot to change its speed instantly, so the low level controller in Fig. 4 is implemented quite straightforwardly. It directly sends the velocity commands and gets the states of the robots via the Player/Stage APIs. The distance d of the handling point is chosen as 0.2 m. The sample period δ is configured as 0.1 s. The weighting matrices in Eq. 17 are chosen as

$$Q = \begin{bmatrix} 0.8 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 0.4 \end{bmatrix}, R = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}. \tag{30}$$

4.1 Validation of the unified delay-free control framework

In this simulation, we aim to demonstrate the effectiveness of the proposed control framework in Fig. 4

Fig. 9 The real-time trajectories of the three robots. The control Algorithm 1 is only adopted by the leader robot and the follower F1, whereas the controller described in Eq. 31 is applied to the follower F2 for comparison



(a) a straight line

(b) a circular way

and Algorithm 1. Three robots are used, with one acting as the leader. the other two robots, indexed as F1 and F2, are the followers who receive state information from the leader. The initial postures of the leader, F1, and F2, are specified as $[0, 0, \pi/4]^T$, $[0, -8.0, 0]^T$, and $[-8.0, 0, \pi/2]^T$. For the follower F1, the desired distance and bearing angle are specified as 1 m and $\pi/6$, while for F2 the desired values are 1 m and $-\pi/6$. For highlighting the novelty of Algorithm 1, only the leader and F1 are deployed with the control framework. Considering the actual situation, the velocity and acceleration constraints for them used in Fig. 5 are assumed to be

$$-2.0 \text{ m/s} \leq v_i \leq 2.0 \text{ m/s},$$

$$-2.0 \text{ rad/s} \leq \omega_i \leq 2.0 \text{ rad/s},$$

and

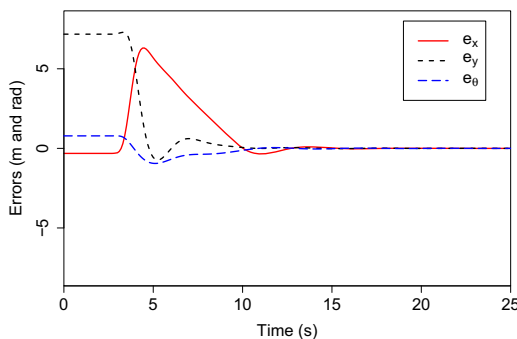
$$-5.0 \text{ m/s}^2 \leq a v_i \leq 5.0 \text{ m/s}^2,$$

$$-5.0 \text{ rad/s}^2 \leq a \omega_i \leq 5.0 \text{ rad/s}^2.$$

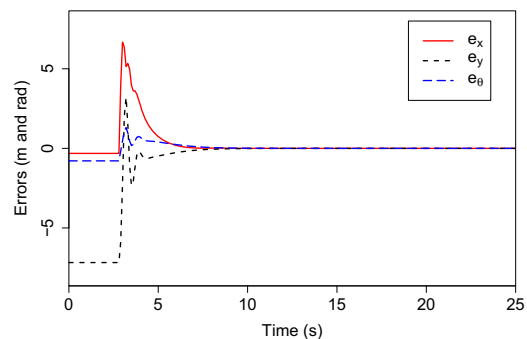
For comparison, the follower F2 is equipped with an ordinary controller based on the Input-Output Linearization [32] as follows:

$$v_i = K1(x_r - x_o) \cos \theta_i + K2(y_r - y_o) \sin \theta_i + v_j + \omega_j \rho_d \sin \phi_d,$$

$$\omega_i = \frac{1}{\rho_d \cos \phi_d} (-K1(x_r - x_o) \sin \theta_i + K2(y_r - y_o) \cos \theta_i - v_j \sin(\theta_i - \theta_j) - \rho_d \sin \phi_d \omega_j \sin(\theta_i - \theta_j)),$$
(31)



(a) the follower F1



(b) the follower F2

Fig. 10 The tracking errors (e_x , e_y , e_θ) of the followers, which move in a straight line

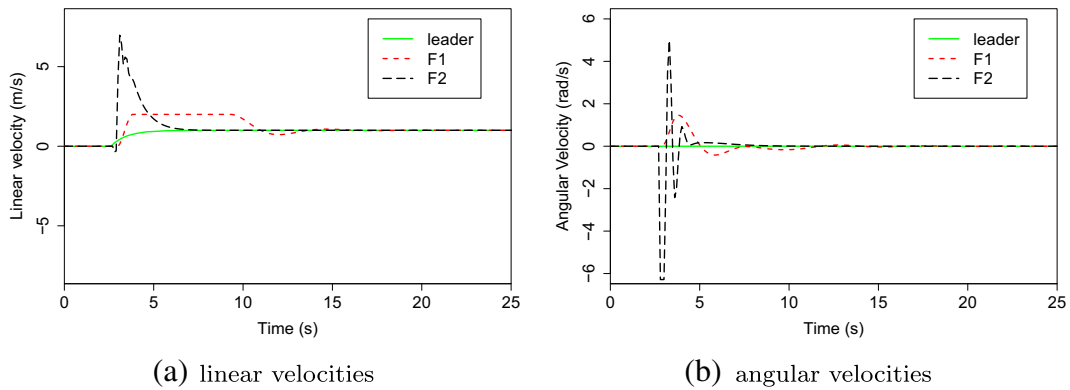


Fig. 11 The linear and angular velocities of the three robots, which move in a straight line. The reference linear and angular velocities of the leader are 1.0 m/s and 0.0 rad/s, respectively

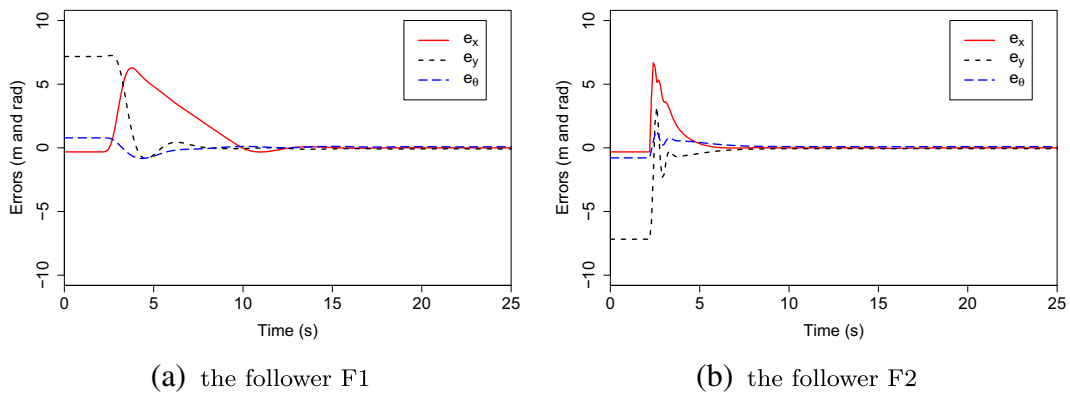


Fig. 12 The tracking errors (e_x , e_y , e_θ) of the followers, which move in a circular way

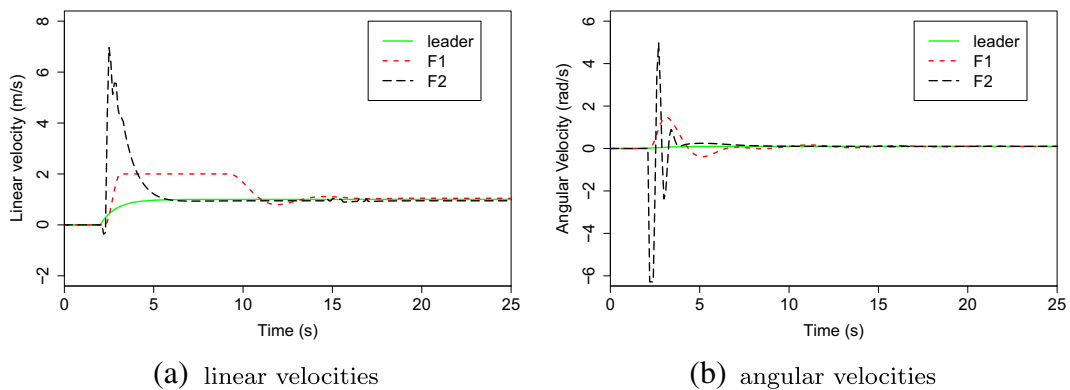


Fig. 13 The linear and angular velocities of the three robots, which move in a circular way. The reference linear and angular velocities of the leader are 1.0 m/s and 0.1 rad/s, respectively

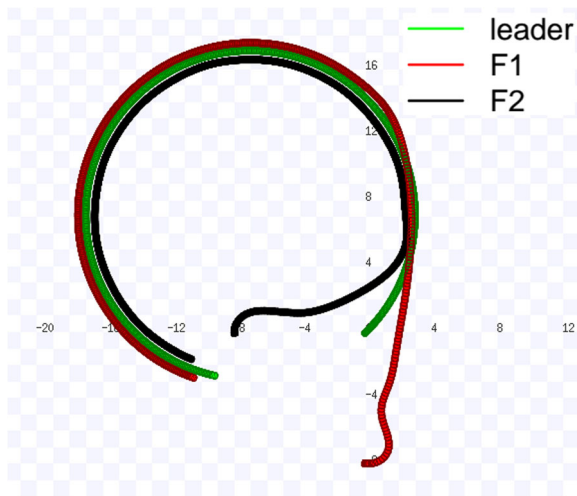


Fig. 14 The real-time circular trajectories of the three robots that are subject to bounded time-varying communication delay

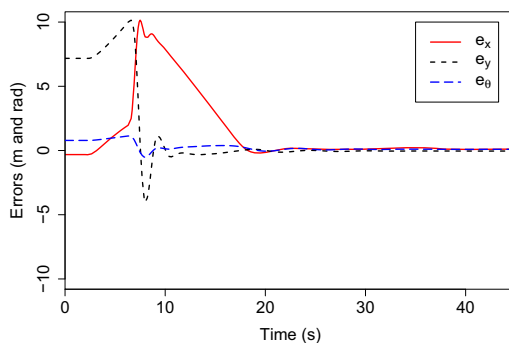
where $K1$ and $K2$ are constants and are selected as $K1 = K2 = 1$, and x_r, y_r, x_o, y_o are calculated by

$$\begin{aligned}
 x_r &= x_j + \rho_d \sin \varphi_d \sin \theta_j, \\
 y_r &= y_j + \rho_d \sin \varphi_d \cos \theta_j, \\
 x_o &= x_i + \rho_d \cos \varphi_d \sin \theta_i, \\
 y_o &= y_i + \rho_d \cos \varphi_d \cos \theta_i.
 \end{aligned}
 \tag{32}$$

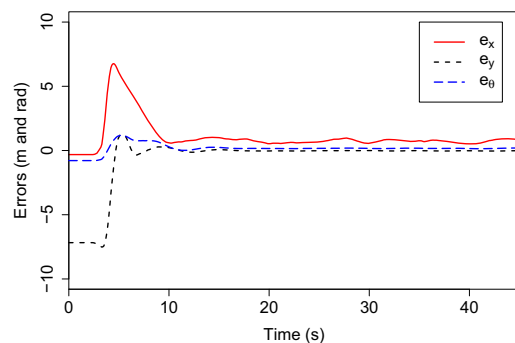
The simulation results from Figs. 9, 10, 11, 12 and 13 show the trajectories, tracking errors, and velocities of the three robots in which the leader is specified to move in a straight line with a fixed linear velocity and angular velocity described as (1.0 m/s, 0.0 rad/s),

as well as to move circularly with the linear and angular velocities (1.0 m/s, 0.1 rad/s). In the simulations, the leader robot starts moving and sends its state information to its followers after a period of time, so we observe constant errors and velocities in the figures at the beginning.

The real-time trajectories in Fig. 9 from the Player/Stage simulation environment verify that the follower F1 is well able to track the leader with both the desired distance and the bearing angle by using the unified leader–follower formation control method proposed in Algorithm 1 that integrates the receding horizon controller. The tracking errors (e_x, e_y, e_θ) shown in Figs. 10 and 12, which converge to zero over time, further confirm the effectiveness of Algorithm 1. It is obvious that the tracking errors of F2 converge much faster than those of F1 in both scenarios, because there are no state and input constraints for the controller (31). As a result, we inevitably observe abrupt changes of the tracking errors of F2 as shown in Figs. 10b and 12b. In contrast, the tracking errors of F1 in Figs. 10a and 12a change smoothly with time, which is more reasonable and more practical. This proves the novelty of Algorithm 1, which is natively capable of taking the constraints into account. The linear and angular velocities shown in Figs. 11 and 13 also reveal the novelty of Algorithm 1. The velocities of F1 always change gradually and smoothly from zero, while the velocity commands generated from the controller (31) suffer impractical jumps from zero to a high value. Overall, the simulation results validate



(a) the follower F1



(b) the follower F2

Fig. 15 The tracking errors (e_x, e_y, e_θ) of the followers, which calculate their velocity commands according to the delayed information from the leader

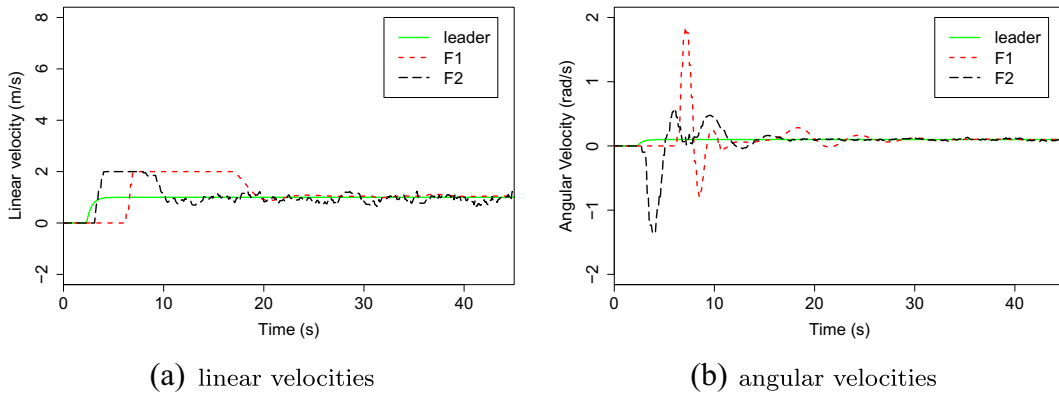


Fig. 16 The linear and angular velocities of the three robots that are subject to communication delay

that the control algorithm 1, combining the receding horizon control principle, is an available and practical solution to the distributed leader–follower formation control problem.

4.2 Validation of the improved control framework subject to bounded time-varying communication delay

This simulation shares the same initial configuration with the previous one. To validate the efficacy of handling bounded time-varying communication delays by using the control framework in Fig. 6 and Algorithm 2, we only apply the improved control framework to the leader and F1, whereas F2 still sticks to Algorithm 1. In other words, F1 will receive delayed predicted future state information from the leader, and F2 will receive delayed current state information. In this simulation, the delays are assumed to be evenly distributed from 0 to 15δ (0 ~ 1.5 s). For F2, if it receives more than one message at a sampling instant, it utilizes the latest one to calculate the tracking error. If no message is received, F2 keeps the previous velocity commands.

Figures 14–16 show the simulation results where the leader is assigned to move in a circular way with the reference linear velocity 1 m/s and angular velocity 0.1 rad/s. Figure 14 shows their trajectories. Since it always receives delayed state information from the leader, we can notice that F2 is a bit behind its desired trajectory from Fig. 14. As a result, the tracking errors of F2 do not converge to zero over time, as

shown in Fig. 15b. Even worse, because of the randomness of the delay, F2 is likely to receive messages out of their sending order, which results in an instability of the formation (see the linear and angular velocities of F2 in Fig. 16). In contrast, the tracking errors of F1 in Fig. 15a show that the improved control framework together with Algorithm 2 solves the leader–follower formation problem that is subject to a time-varying communication delay. The linear and angular velocities of F1 in Fig. 16 also indicate that the improved control framework is able to keep the formation stable.

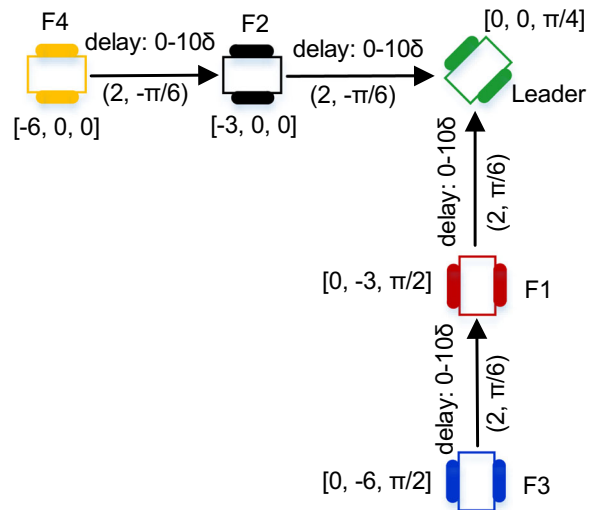
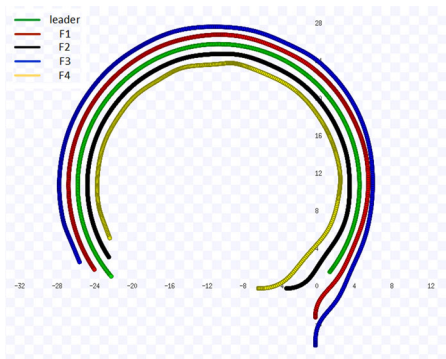
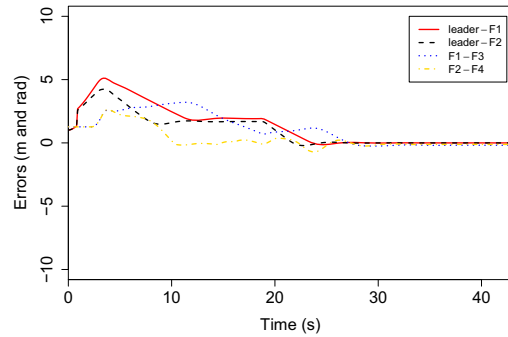


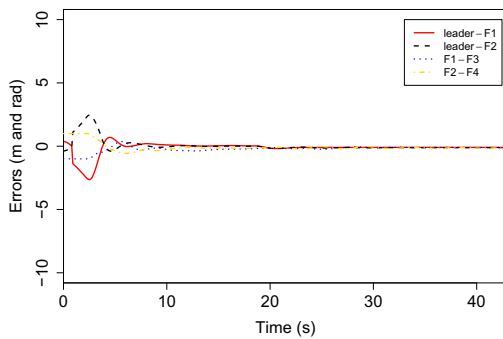
Fig. 17 Initial postures and formation configuration of the three robots



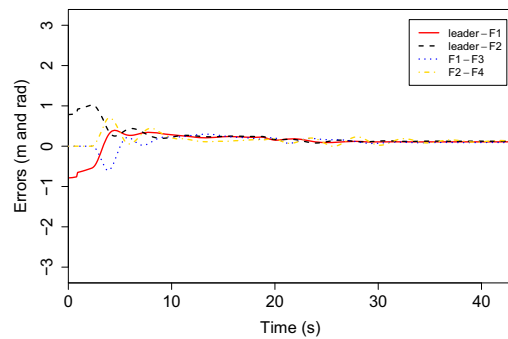
(a) The real-time trajectories of five robots that have different internal clocks.



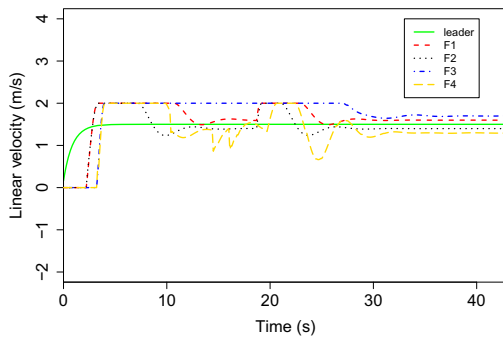
(b) tracking error e_x



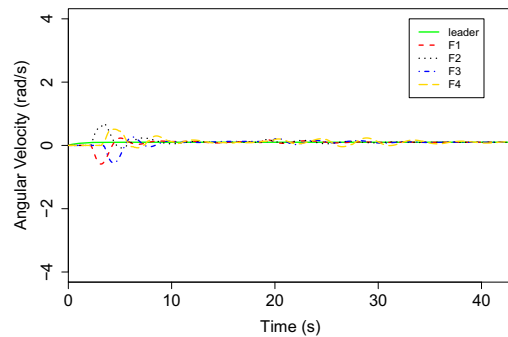
(c) tracking error e_y



(d) tracking error e_θ



(e) linear velocities



(f) angular velocities

Fig. 18 The simulation results of five robots that are subject to time-varying communication delays and asynchronous clocks

4.3 Validation of the control framework subject to asynchronous clocks

To clearly verify the significance of the syn-point method when coping with the asynchronous clock

problem among robots, this simulation employs five robots to form a shape, but is slightly different from the previous two simulations. The initial postures and formation configuration are shown in Fig. 17. Compared with the previous two simulations, their

initial postures are configured to clearly show the tracking errors. F1 and F2 are assigned to track the leader robot in this simulation, while F3 and F4 are assigned to track F1 and F2, respectively. For F1 and F3, the desired distance and bearing angle are 2.0 m and $\pi/6$ rad. For F2 and F4, the desired distance and bearing angle are 2.0 m and $-\pi/6$ rad. The communication delay for all leader–follower pairs is assumed to be time-varying from 0 to 10δ ($0 \sim 1.0$ s). The reference linear and angular velocities for the leader robot are 1.5 m/s and 0.1 rad/s, respectively. All the robots are equipped with the distributed control framework described in Fig. 6 and Algorithm 3.

Figure 18 presents the simulation results. Their real-time trajectories in Fig. 18a show the effectiveness of Algorithm 3 at dealing with bounded time-varying communication delays and the asynchronous clock problem. Moreover, their tracking errors, shown in Fig. 18b, c, and d, which converge to zero over time, confirm the feasibility and effectiveness of Algorithm 3 that adopts the syn-point based approach. Figure 18e shows that their linear velocities start from zero, and converge to the desired values gradually. The angular velocities of F1, F2, F3, and F4, shown in Fig. 18f, converge to the angular velocity of the leader gradually.

Let us further analyse Fig. 18b and e to see how Algorithm 3 works. The leader robot starts to move at time 0.0 s, at which time it also sends its predicted state information to F1 and F2. Due to the asynchronicity of clocks, F1 and F2 will not know the exact delay of the first message from the leader. Algorithm 3 always assumes that the delay of the first message is zero, so it will be stored in a position with $2 * \bar{D}$ (2.0 s) delay compared to the current position. As a result, F1 and F2 start to move and send messages to F3 and F4 at time 2.3 s. Analogously, F3 and F4 starts to move at time 3.3 s. The tracking errors of F1 and F2 become nearly stable at times 11.7 s and 10.1 s. However, their tracking errors cannot converge to zero completely until a syn point is detected. F1 and F2 both finally detect a syn point at times 18.8 s, and then they begin to speed up to catch up with the leader. For F3 and F4, we do not observe the effects of the syn-point based approach from the figures, as the syn points are detected when they are both in full speed during $0 \sim 10$ s. At time 30.3 s, the desired formation is achieved, and becomes stable after that.

5 Conclusion and Future Work

In this paper, based on the receding horizon control principle, we have implemented a fully distributed control framework integrating several techniques to solve the leader–follower formation control problem for a group of networked nonholonomic mobile robots that have asynchronous clocks and suffer from bounded time-varying communication delays. The framework can be deployed indiscriminately on both the leader and the follower sides, to make them converge to and move as a desired formation. By adding an auxiliary acceleration term to the receding horizon controller, the framework solves the impractical velocity jump problem, as well as meets the state and input constraints. To solve the communication delay problem, we propose an improved control framework that deliberately utilizes the predictability of the receding horizon controller. As the messages may be received out of order on the follower side, a novel technique that combines the timing and order features of messages is exploited to guarantee the logical correctness of the messages. We carefully analysed the characteristics of a bounded time-varying communication delay, and propose a syn point based method to solve the asynchronous clock problem, which can be natively integrated into the framework. As a result, there is no need to integrate a delicate time synchronizing protocol into the robots in a fully distributed environment. The simulation results demonstrate the effectiveness of the proposed approaches.

The control framework proposed in this paper employs simplified models for the world and the robots, and only focusses on the bounded time-varying delay and asynchronous clock problems. However, there are some issues that should be further considered, such as communication losses and obstacles avoidance. Communication losses could happen inevitably, for example due to block and congestion of the communication channel. The control stability conditions and laws subject to communication losses need to be carefully derived, especially under the scheme of receding horizon control. In a real environment, the working space is usually scattered with obstacles that could alter the trajectories of the robots. Therefore, some obstacle avoidance mechanisms should be integrated into the receding horizon control scheme. In future work, we will investigate these issues, and

give a more practical solution to the formation control problem.

Acknowledgments This work was supported by the China Scholarship Council (CSC) under Grant No. 201206110039.

References

- Balch, T., Arkin, R.C.: Behavior-based formation control for multirobot teams. *IEEE T. Robot. Autom.* **14**(6), 926–939 (1998)
- Chaimowicz, L., Cowley, A., Gomez-Ibanez, D., Grocholsky, B., Hsieh, M., Hsu, H., Keller, J., Kumar, V., Swaminathan, R., Taylor, C.: Deploying air-ground multi-robot teams in urban environments. In: *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, pp. 223–234. Springer (2005)
- Chen, H., Allgower, F.: A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. In: *1997 European Control Conference (ECC)*, pp. 1421–1426. IEEE (1997)
- Chen, J., Sun, D., Yang, J., Chen, H.: A leader-follower formation control of multiple nonholonomic mobile robots incorporating receding-horizon scheme. *International Journal of Robotics Research* (2009)
- Cheng-Lin, L., Yu-Ping, T.: Formation control of second-order dynamic agents with heterogeneous communication delays. In: *2008 27th Chinese Control Conference*, pp. 536–540. IEEE (2008)
- Collett, T.H., MacDonald, B.A., Gerkey, B.P.: Player 2.0: Toward a practical robot programming framework. In: *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005)*, p. 145 (2005)
- Dai, Y., Lee, S.G.: The leader-follower formation control of nonholonomic mobile robots. *Int. J. Control Autom.* **10**(2), 350–361 (2012)
- D’Andréa-Novel, B., Bastin, G., Campion, G.: Modelling and control of non-holonomic wheeled mobile robots. In: *1991 IEEE International Conference on Robotics and Automation*, pp. 1130–1135. IEEE (1991)
- Das, A.K., Fierro, R., Kumar, V., Ostrowski, J.P., Spletzer, J., Taylor, C.J.: A vision-based formation control framework. *IEEE T. Robot. Autom.* **18**(5), 813–825 (2002)
- Defoort, M., Floquet, T., Kokosy, A., Perruquetti, W.: Sliding-mode formation control for cooperative autonomous mobile robots. *IEEE T. Ind. Electron.* **55**(11), 3944–3953 (2008)
- Dong, X., Xi, J., Lu, G., Zhong, Y.: Formation control for high-order linear time-invariant multiagent systems with time delays. *IEEE Trans. Control Netw. Syst.* **1**(3), 232–240 (2014)
- Dunbar, W.B., Murray, R.M.: Distributed receding horizon control for multi-vehicle formation stabilization. *Automatica* **42**(4), 549–558 (2006)
- Findeisen, R., Imsland, L., Allgower, F., Foss, B.A.: State and output feedback nonlinear model predictive control: An overview. *Eur. J. Control* **9**(2), 190–206 (2003)
- Franzè, G., Lucia, W.: An obstacle avoidance model predictive control scheme for mobile robots subject to nonholonomic constraints: A sum-of-squares approach. *J. Frankl. Inst.* **352**(6), 2358–2380 (2015)
- Gerkey, B., Vaughan, R.T., Howard, A.: The player/stage project: Tools for multi-robot and distributed sensor systems. In: *Proceedings of the 11th international conference on advanced robotics*, vol. 1, pp. 317–323 (2003)
- Gustavi, T., Hu, X.: Observer-based leader-following formation control using onboard sensor information. *IEEE T. Robot.* **24**(6), 1457–1462 (2008)
- Hougen, D.F., Benjaafar, S., Bonney, J.C., Budenske, J.R., Dvorak, M., Gini, M., French, H., Krantz, D.G., Li, P.Y., Malver, F., et al.: A miniature robotic system for reconnaissance and surveillance. In: *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 501–507. IEEE (2000)
- Houska, B., Ferreau, H.J., Diehl, M.: An auto-generated real-time iteration algorithm for nonlinear mpc in the microsecond range. *Automatica* **47**(10), 2279–2285 (2011)
- Izadi, H.A., Gordon, B.W., Zhang, Y.: Decentralized receding horizon control for cooperative multiple vehicles subject to communication delay. *J. Guid., Control, Dyn.* **32**(6), 1959–1965 (2009)
- Izadi, H.A., Gordon, B.W., Zhang, Y.: Hierarchical decentralized receding horizon control of multiple vehicles with communication failures. *IEEE T. Aero. Elec. Sys.* **49**(2), 744–759 (2013)
- Jiang, L., Zhang, R.: Stable formation control of multi-robot system with communication delay, vol. 9 (2012)
- Lewis, M.A., Tan, K.H.: High precision formation control of mobile robots using virtual structures. *Auton. Robot.* **4**(4), 387–403 (1997)
- Li, X., Xiao, J., Cai, Z.: Backstepping based multiple mobile robots formation control. In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 887–892. IEEE (2005)
- Liu, J., Muñoz de la Peña, D., Christofides, P.D., Davis, J.F.: Lyapunov-based model predictive control of nonlinear systems subject to time-varying measurement delays. *Int. J. Adapt. Control* **23**(8), 788–807 (2009)
- Münz, U., Papachristodoulou, A., Allgöwer, F.: Delay-dependent rendezvous and flocking of large scale multi-agent systems with communication delays. In: *2008 47th IEEE Conference on Decision and Control*, pp. 2038–2043. IEEE (2008)
- Nourbakhsh, I.R., Sycara, K., Koes, M., Yong, M., Lewis, M., Burion, S.: Human-robot teaming for search and rescue. *IEEE Pervas. Comput.* **4**(1), 72–79 (2005)
- Olfati-Saber, R., Murray, R.M.: Consensus problems in networks of agents with switching topology and time-delays. *IEEE T. Automat. Contr.* **49**(9), 1520–1533 (2004)
- Peng, Z., Wen, G., Rahmani, A., Yu, Y.: Leader-follower formation control of nonholonomic mobile robots based on a bioinspired neurodynamic based approach. *Robot. Auton. Syst.* **61**(9), 988–996 (2013)

29. Ren, W., Beard, R.: Decentralized scheme for spacecraft formation flying via the virtual structure approach. *J. Guid. Control Dyn.* **27**(1), 73–82 (2004)
30. Sadowska, A., Den Broek, T.V., Huijberts, H., van de Wouw, N., Kostić, D., Nijmeijer, H.: A virtual structure approach to formation control of unicycle mobile robots using mutual coupling. *Int. J. Control* **84**(11), 1886–1902 (2011)
31. Sanchez, J., Fierro, R.: Sliding mode control for robot formations. In: 2003 IEEE International Symposium on Intelligent Control, pp. 438–443. IEEE (2003)
32. Shao, J., Xie, G., Wang, L.: Leader-following formation control of multiple mobile vehicles. *IET Control Theory Appl.* **1**(2), 545–552 (2007)
33. Sipahi, R., Niculescu, S.I., Abdallah, C.T., Michiels, W., Gu, K.: Stability and stabilization of systems with time delay. *IEEE Contr. Syst. Mag.* **31**(1), 38–65 (2011)
34. Stipanović, D.M., Inalhan, G., Teo, R., Tomlin, C.J.: Decentralized overlapping control of a formation of unmanned aerial vehicles. *Automatica* **40**(8), 1285–1296 (2004)
35. Stoeter, S.A., Rybski, P.E., Stubbs, K.N., McMillen, C.P., Gini, M., Hougen, D.F., Papanikolopoulos, N.: A robot team for surveillance tasks: Design and architecture. *Robot Auton. Syst.* **40**(2), 173–183 (2002)
36. Sun, B., Zhu, D., Yang, S.X.: A bioinspired filtered backstepping tracking control of 7000-m manned submarine vehicle. *IEEE Trans. Ind. Electron.* **61**(7), 3682–3693 (2014)
37. Tian, Y.P., Zhang, Y.: High-order consensus of heterogeneous multi-agent systems with unknown communication delays. *Automatica* **48**(6), 1205–1212 (2012)
38. Vidal, R., Shakerinia, O., Sastry, S.: Formation control of nonholonomic mobile robots with omnidirectional visual servoing and motion segmentation. In: 2003 IEEE International Conference on Robotics and Automation, vol. 1, pp. 584–589. IEEE (2003)
39. Werfel, J., Petersen, K., Nagpal, R.: Designing collective behavior in a termite-inspired robot construction team. *Science* **343**(6172), 754–758 (2014)
40. Xu, D., Zhang, X., Zhu, Z., Chen, C., Yang, P.: Behavior-based formation control of swarm robots, vol. 2014 (2014)
41. Xu, Z., Schröter, M., Neacsulescu, D., Ma, L., Schilling, K.: Formation control of car-like autonomous vehicles under communication delay. In: 2012 31st Chinese Control Conference, pp. 6376–6383. IEEE (2012)
42. Yoon, Y., Shin, J., Kim, H.J., Park, Y., Sastry, S.: Model-predictive active steering and obstacle avoidance for autonomous ground vehicles. *Control Eng. Pract.* **17**(7), 741–750 (2009)
43. Yoshioka, C., Namerikawa, T.: Formation control of nonholonomic multi-vehicle systems based on virtual structure. In: 17th IFAC World Congress, pp. 5149–5154 (2008)

Long Peng received his BS and MS in Computer Science and Technology from the National University of Defense Technology, China in 2009 and 2012. His areas of research were web programming, data mining, distributed system. Since Nov. 2012 he has been awarded as a Ph.D. candidate at the Embedded System Lab, ETRO, VUB. His research topics include real-time operating system benchmark, behavior-based robotics and adaptive systems.

Fei Guan received her B.Eng degree in Automatic Test and Control from Harbin Institute of Technology (HIT), China, in 2010. She received her M.S. degree at the same university in 2012 and is currently a Ph.D. Candidate at the Department of Electronics and Informatics (ETRO) of Vrije Universiteit Brussel (VUB). Her research interests are in the areas of real-time operating system and adaptive scheduling.

Luc Perneel received his master degree in Electrical Engineering from Catholic University Leuven (KUL), Belgium, in 1990. This was complemented with a post master degree in Industrial Management, awarded also by KUL, in 1991. He has more than twenty years of experience in embedded systems and real-time behavior. He is specialized in Operating Systems and interactions between hardware and software. Besides performing research on real-time scheduling at VUB as a PhD candidate, he owns his company “Luperco”.

Hasan Fayyad-Kazan received his Master degree in Applied Computer Science from the Vrije Universiteit Brussel (VUB) in 2010. He did his Ph.D. afterwards in VUB too, at the EmSlab. Hasan finished his Ph.D. in 2014, which was about Benchmarking virtualization solutions for both business and real-time systems. During his studies, Hasan worked at Dedicated Systems Experts in building and testing real-time operating systems. He is currently working as a postdoc. Hasan’s specialities are: Embedded systems, real-time systems, virtualization and cloud computing.

Martin Timmerman obtained his Engineering Master in Telecommunications at the Royal Military Academy, Brussels in 1975 and the Ph.D. degree in applied science at the University of Gent in 1982. He is a professor at the Royal Military Academy and VUB. His research focuses on real-time operating system and swarm robotics. He is the head of the Embedded Systems Lab at VUB.