IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 27, 2020, accepted August 9, 2020, date of publication August 13, 2020, date of current version August 25, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3016347

Dynamic Task Priority Planning for Null-Space Behavioral Control of Multi-Agent Systems

YUTAO CHEN, ZHENYI ZHANG, (Student Member, IEEE), AND JIE HUANG, (Member, IEEE)

College of Electrical Engineering and Automation, Fuzhou University, Fuzhou 350108, China

Corresponding author: Jie Huang (jie.huang@fzu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61603094.

ABSTRACT The task priority planning problem is addressed in the task supervisor of null-space behavioral (NSB) control for multi-agent systems. Traditional methods rely on pre-defined logic-based or fuzzy rules to adjust task priority. In this work, a novel task supervisor is proposed using model predictive control (MPC) techniques. At each sampling instant, the task priority planning problem is formulated as a switching mode optimal control problem (OCP), which can be solved by efficient mixed-integer optimal control algorithms. The optimal task priority order is obtained based on current and predictive information of agents, without the need for a pre-defined rule. By explicitly introducing slack variables into constraints, the proposed MPC method is flexible to cope with dynamic obstacles in unknown environments. Simulations with static and dynamic obstacles show that the proposed method can provide significantly better control performance than the traditional logic-based method using less priority switchings.

INDEX TERMS Multi-agent systems, behavioral control, task supervisor, model predictive control.

I. INTRODUCTION

Because of their agility and versatility, multi-agent systems are widely applied in both military [1], [2] and civilian areas [3]–[5]. Theoretic research on multi-agent systems have also been widely addressed [6], [7]. Autonomous mobile robots are typical multi-agent systems which usually work in an unstructured environment and need to accomplish multiple tasks such as moving through predefined via-points, avoiding static or dynamic obstacles, forming a formation and flocking. However, these tasks may conflict, e.g. an autonomous mobile robot cannot move through a predefined via-point while at the same time avoid an obstacle near the via-point.

The behavioral control, first proposed by Brooks [8], is one popular control method to handle task conflicts and has been studied intensively in the past decade. The traditional methodology of the behavioral control mainly includes the layered control approach [8] and the motor schema control approach [9], [10]. The layered control approach is a competitive approach in which tasks with a lower priority can only be executed after those with a higher priority have been completed. The motor schema control is a cooperative approach in which the output of each task is summed up with weights to generate the final task output. In this way, no task is completely achieved. More recently, Antonelli and Chiaverini [11] has proposed a cooperative approach named null-space-based behavioral (NSB) control to solve task conflicts by assigning a priority to each task and designing a priority-based task coordination scheme. The NSB method guarantees that tasks with a higher priority can be executed completely on one hand, and on the other hand tasks with a lower priority are partially executed via null-space projection. Therefore, the NSB method is potential in nature in handling task conflicts. However, the priority of tasks, which is determined by the task supervisor, is usually set in advance and is fixed during the entire task execution process [12], [13]. This may degrade the control performance of the NSB method and limit its application in dynamic and unknown environments. For instance, setting the obstacle-avoidance task as the top priority when obstacles are not present would affect the performance of other tasks.

Some efforts have been made to tackle this problem. Finite state automata (FSA) is a state transfer mechanism in which system states automatically transfer from one to another based on system current state and state transition trigger conditions. In the context of NSB control, the planning of

The associate editor coordinating the review of this manuscript and approving it for publication was Jianxiang Xi.

task priority is realized by designing the rules for state transition trigger conditions [14]–[17]. Alternatively, task priority planning has been studied by designing certain fuzzy rules using the fuzzy method for autonomous systems [15], [18], [19]. However, all aforementioned methods require human to design the planning rules for task priority planning in advance. This is difficult when the number of tasks is large and the environments are dynamic and unknown. In addition, the planning of tasks priority, either using FSA or the fuzzy method, only makes use of the current state of the agents without future predictions.

It should be noted that advanced priority planning or behavior switching algorithms have been studied in the field of robots and multi-agent systems but not in the NSB control framework. In [20], a model predictive control (MPC) algorithm has been proposed to compute the switching time of robot behaviors, given a known switching order in advance. In [21] and [22], genetic algorithms have been exploited for robot task switching and behavior selection, respectively.

In this paper, a novel task supervisor to dynamically plan task priority in the NSB control is proposed using MPC techniques. MPC as an advanced controller, has been employed in robot and multi-agent systems mainly for trajectory tracking and obstacle avoidance [23], [24]. Here, MPC works as a high-level planner to provide dynamically the priority of tasks. First, different task priority orders formulate multiple composite tasks. Then, at each sampling instant, a switching mode optimal control problem (OCP) is formulated by explicitly taking into account system dynamics and constraints. The optimal composite task trajectory is obtained by solving the OCP making use of current state and future predictions. As a result, task priority planning can be realized on-line without a pre-defined adjustment rule. Finally, a simulation with static and dynamic obstacles is shown to demonstrate the effectiveness of the proposed method. It can be found that the predictive nature of the proposed method leads to earlier and smoother switching compared to the traditional logic-based method.

This paper is organized as follows. Section II briefly introduces preliminaries of the NSB control and the basic problem description. In Section III, the MPC based task priority planner, which is the main contribution of the paper, is presented. Section IV presents in detail a simulation in two scenarios. Finally, a conclusion is drawn.

II. PRELIMINARIES AND PROBLEM DESCRIPTION

A. ELEMENTARY TASK

In the NSB control, elementary tasks can be organized in different orders. Then, tasks fusion is designed to obtain different composite tasks with a priority order. Note that the term ''task'' and ''behavior'' have the same meaning in this paper, because the ''behavior'' is also named task or mission in the behavioral control [13]. An elementary task is encoded by a task variable $\rho \in \mathbb{R}^m$, which is a function of the system configuration $x \in \mathbb{R}^n$, expressed as

$$\rho = g(x), \tag{1}$$

with the corresponding derivative:

$$\dot{\rho} = \frac{\partial g(x)}{\partial x} v = J(x)v, \tag{2}$$

where $J(x) \in \mathbb{R}^{m\times n}$ is a configuration-dependent task Jacobian matrix and $v \in \mathbb{R}^n$ is the stacked vector of the agents' velocities. By inverting the locally linear mapping (2) into thee least square formulation (LSS), the reference velocity v_d can be computed by:

$$v_d = J^{\dagger}\dot{\rho}_d = J^T(JJ^T)^{-1}\dot{\rho}_d, \tag{3}$$

where ρ_d is the reference position, $J^{\dagger} = J^T(JJ^T)^{-1} \in \mathbb{R}^{n\times m}$ is the pseudo-inverse of Jacobian matrix. In practice, the integration of the reference velocity would result in a numerical drift of the reconstructed agent's position. The following closed-loop inverse kinematics (CLIK) form has been used to compensate such drift:

$$v_d = J^{\dagger}(\dot{\rho} + \Lambda\tilde{\rho}), \tag{4}$$

where Λ is a suitable constant positive-definite matrix of gains and $\tilde{\rho} = \rho_d - \rho$ is the task error.

B. COMPOSITE TASK

A composite task is a combination of multiple elementary tasks arranged in a prioritized order. Let $\rho_b \in \mathbb{R}^{m_b}$ be the task function, where $b \in \mathbb{N}_b$, $\mathbb{N}_b = \{1, \ldots, h\}$, and m_b is the dimension of the b-th task space. Define a time-dependent priority function $y(b, t) : \mathbb{N}_b \times [0, \infty] \to \mathbb{N}_b$ that maps the task function space to a priority space. Also define a task hierarchical structure which complies with the following rules [25]:

1) A task b of priority $y(b, t)$ at time t must not disturb task c of priority $y(c, t)$ if $y(b, t) \geq y(c, t)$, $\forall b, c \in \mathbb{N}_b$, $b \neq c$.
2) The mappings from the velocities to the task velocities are captured by the task Jacobian matrix $J_{y(b,t)} \in \mathbb{R}^{m_b\times n}$, $b \in \mathbb{N}_b$.
3) The dimension of task b with the lowest priority may be larger than $n - \sum_{c=1}^{r|c\neq b} m_c$ so that the dimension n is ensured to be greater than the total dimension of all tasks.
4) The value of $y(b, t)$ is assigned by the task supervisor on the basis of the mission's need and sensor information.

By assigning elementary tasks with a given priority, the composite task velocity at time t can be expressed as

$$v_d(t) = v_1 + (I - J_1^{\dagger}J_1(v_2 + (I - J_2^{\dagger}J_2(v_3 \tag{5}$$

$$+ \ldots + (I - J_{n-1}^{\dagger}J_{n-1})v_n))), \tag{6}$$

where v_b, $b = 1, 2, \ldots, h$ is the velocity of the task c with priority $y(c, t) = b$.

C. TASK SUPERVISOR

A task supervisor is in charge of switching between the defined composite tasks based on the current state of agents. Assume that there are M composite tasks which are composed of the same elementary tasks arranged in different

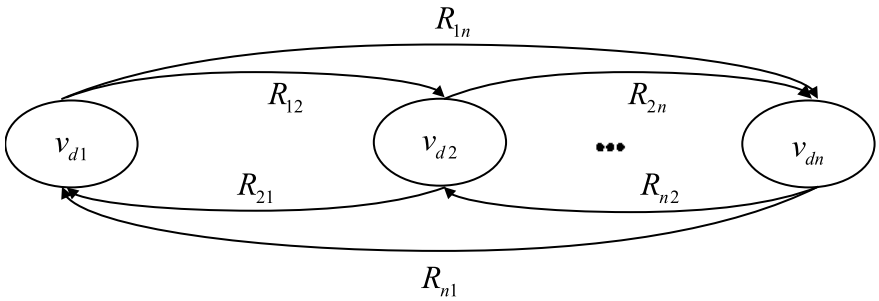


FIGURE 1. Principle block diagram of a conventional logic-based task supervisor.

prioritized orders. Let v_{db}, $b = 1, \ldots, M$ represent the velocity of the $b-$th composite task. The principle block diagram of a conventional task supervisor is shown in Fig. 1, in which R_{bc}, $b = 1, \ldots, M$, $c = 1, \ldots, M$, $b \neq c$ represent the task switching trigger conditions designed by human based on the state of agents. Once the task switching trigger conditions are satisfied, the composite task is switched to another, resulting in adjustment of task priority. An example of the traditional task supervisor is to achieve priority switching between the motion task and the obstacle-avoidance task, in which the task switching trigger conditions are designed based on logic rules, i.e. if the distance between an agent and an obstacle is greater than or equal to a safe distance, the obstacle-avoidance task has a higher priority, otherwise the motion task has a higher priority.

III. DYNAMIC TASK PRIORITY PLANNING

The traditional NSB method usually has a pre-defined and fixed priority order during the entire task execution process, i.e. y is a time-independent function. In this paper, a task priority planning method is proposed using model predictive control techniques.

A. OPTIMAL CONTROL FORMULATION

Consider a group of n ($n \geq 2$) agents, and a set of $h (h \geq 2)$ tasks for each agent. Define the state vector $\mathbf{x}$ as $\mathbf{x}(t) = [x_1(t)^\top, \ldots, x_n(t)^\top]^\top$, each element of which represents the state of an agent. Let us introduce a binary mode vector $\mathbf{w}(t) = [w_1(t)^\top, \ldots, w_n(t)^\top]^\top \in \{0, 1\}^{nM}$, each element $w_i(t) \in \{0, 1\}^M$. The dynamics of an agent can be written as a convex combination of dynamics in different modes:

$$\dot{x}_i(t) = \sum_{j=1}^{M} w_i^j(t) v_{d,i}^j(t), \quad i = 1, \ldots, n \tag{7}$$

where w_i^j is the jth element of w_i and $v_{d,i}^j$ is the jth task velocity in the form of (5) of the ith agent. The total number of task velocities M is computed by $M = h!$, the factorial of the number of tasks h. This is because h tasks can generate $h!$ composite tasks with elementary task in different prioritized orders. An important constraint of $\mathbf{w}$ is

$$\sum_{j=1}^{M} w_i^j(t) = 1, \quad i = 1, \ldots, n, \tag{8}$$

which is called the special-ordered-set-of-type-one (SOS1) [26], ensuring at least but only one active mode at any time instant.

Define the tracking error for the ith agent as

$$e_i(t) = \rho_d^i - x_i(t), \tag{9}$$

where ρ_d^i is the reference position for the ith agent. Define a cost function

$$L(\mathbf{x}, \mathbf{u}) = \int_0^T \sum_{i=1}^{n} \left(K_i \|e_i(t)\|^2 + P_i \|u_i(t)\|^2 \right) \mathrm{d}t, \tag{10}$$

which consists of combined tracking errors from all agents plus slack variables u_i, $i = 1, \ldots, n$, each weighted by positive parameters K_i and P_i. As a result, an OCP can be formulated as

$$\min_{\mathbf{x}, \mathbf{w}, \mathbf{u}} \; L(\mathbf{x}) \tag{11a}$$

$$s.t. \; \mathbf{x}(0) = \hat{x}_0 \tag{11b}$$

$$(7), (8),$$

$$\|x_i(t) - x_O^i(t)\| \geq d_i - u_i(t), \quad i = 1, \ldots, n, \quad \forall t \in [0, T] \tag{11c}$$

$$u_i(t) \geq 0, \quad \forall t \in [0, T], \tag{11d}$$

where $\hat{x}_0$ is the initial sate, $x_O^i(t)$ is the obstacle position detected by the ith agent at time t, d_i is the safety distance lower bound for the ith agent, and T is the prediction horizon. The slack vector is $\mathbf{u}(t) = [u_1(t)^\top, \ldots, u_n(t)^\top]^\top$. Constraint (11d) ensures that all slack variables are non-negative, making (11c) a soft constraint in case of local minimum problems.

Remark 1: In case of dynamic obstacles, agents may repeatedly violate the safe distance constraint in multiple sampling instants even though they have already set the obstacle-avoidance task as the top priority one. This case is more complicated than the traditional local minimum problem where agents stop moving because of two opposite velocities from different tasks. For the traditional method, additional algorithms, such as adding small measurement noises [25] and introducing human intervention [27], are required to handle the local minimum problem. However, the proposed MPC method in the form of (11) can handle dynamic obstacles and local minimum problems explicitly and systematically by incorporating constraint (11c) and (11d). In this way, agents can temporarily violate the safety distance constraint in an optimal way depending on current situation and return to other behaviors when possible.

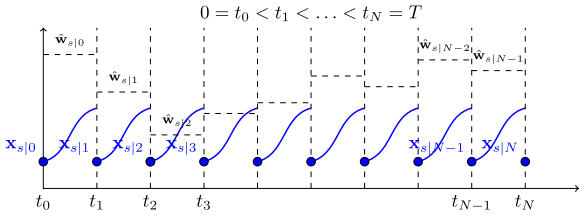


FIGURE 2. An illustration of the direct discretization method with N initial guesses of $\mathbf{X}$ and $N-1$ initial guesses of $\hat{\mathbf{W}}$.

B. REAL-TIME MODEL PREDICTIVE CONTROL ALGORITHM

During the task execution process, problem (11) must be solved repeatedly with different initial states $\hat{x}_0$ updated at every sampling instant, leading to a model predictive control (MPC) framework. However, problem (11) contains integer variables which is difficult to solve. In this paper, we employ a real-time MPC algorithm [26] that can efficiently solve (11) on-line, providing a dynamic task priority planning. The algorithm employs a direct method using "first discretize then optimize" methodology, and an outer-convexfication with integer relaxation technique.

Divide the domain $[0, T]$ into N intervals, characterized by equidistant grid points

$$0 = t_0 < t_1 < \ldots < t_{N-1} < t_N = T,$$

with $\Delta t = \frac{T}{N}$. In each interval, the binary variables $\mathbf{w}$ are assumed to be constant hence they can only change values at grid points. The binary variables $\mathbf{w}$ are relaxed to real variables satisfying $\hat{\mathbf{w}}(t) \in [0, 1]^{nM}$. Using multiple shooting [28], problem (11) can be converted to a nonlinear programming problem (NLP), written as

$$\min_{\mathbf{X},\hat{\mathbf{W}},\mathbf{U}} \sum_{k=0}^{N} L(\mathbf{x}_{s|\mathbf{k}}, \mathbf{u}_{s|\mathbf{k}}) \tag{12a}$$

$$s.t.\ \mathbf{x}_{s|0} = \hat{x}_0 \tag{12b}$$

$$\mathbf{x}_{s|k+1} = \phi(\mathbf{x}_{s|k}, \hat{\mathbf{w}}_{s|k}), \quad k = 0, \ldots, N-1 \tag{12c}$$

$$\sum_{j=1}^{M} w^j_{i,s|k} = 1, \quad i = 1, \ldots, n, \quad k = 0, \ldots, N-1, \tag{12d}$$

$$\|x_{i,s|k} - x^i_O\| \geq d_i - u_{i,s|k}, \quad i = 1, \ldots, n, \quad k = 0, \ldots, N, \tag{12e}$$

$$u_{i,s|k} \geq 0, \quad i = 1, \ldots, n, \quad k = 0, \ldots, N, \tag{12f}$$

where

$$\mathbf{X} = (\mathbf{x}^\top_{0|0}, \mathbf{x}^\top_{0|1}, \ldots, \mathbf{x}^\top_{0|N})^\top,$$
$$\hat{\mathbf{W}} = (\hat{\mathbf{w}}^\top_{0|0}, \hat{\mathbf{w}}^\top_{0|1}, \ldots, \hat{\mathbf{w}}^\top_{0|N-1})^\top,$$
$$\mathbf{U} = (\mathbf{u}^\top_{0|0}, \mathbf{u}^\top_{0|1}, \ldots, \mathbf{u}^\top_{0|N})^\top,$$

and s is the current sample, $\mathbf{x}_{s|k}$ is the $k-$step predicted state at sample s, $x_{i,s|k}$ is ith element of $\mathbf{x}_{s|k}$ given by (7) and $u_{i,s|k}$ is ith slack variable at sample s. An illustration of this discretization at the initial guess is shown in Fig. 2. In (12), the function $\phi : \mathbb{R}^{nn_x} \times [0, 1]^{nM}$ is obtained by solving the nonlinear dynamics (7) via numerical integration, e.g. the Euler method. Constraint (12c) ensures that the state trajectory is continuous over the entire prediction horizon at the solution of (12) after convergence is achieved. An illustration of the state and mode trajectory at the solution is shown in Fig. 3. The NLP (12) can be solved efficiently by standard NLP solvers using sequential quadratic programming or interior point methods, without using integer optimization algorithms.

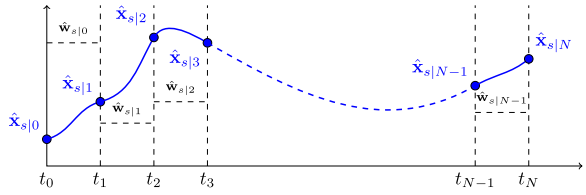


FIGURE 3. The continuous trajectory at the solution of (12) after convergence is achieved.

After solving (12), a sum-up-rounding (SUR) step can be employed to obtain the binary variable $\mathbf{W}$ from $\hat{\mathbf{W}}$ [26]. The SUR step reads as

$$g^j_{s|k} = \sum_{r=0}^{k} \hat{w}^j_{s|r} \Delta t - \sum_{r=0}^{k-1} w^j_{s|r} \Delta t, \tag{13a}$$

$$w^j_{s|k} = \begin{cases} 1 & \text{if } g^j_{s|k} \geq g^r_{s|k} \quad \forall r \neq j \& j < r \ \forall r : g^j_{s|k} = g^r_{s|k} \\ 0 & \text{otherwise,} \end{cases} \tag{13b}$$

for $j = 1, \ldots, M$ and $k = 0, \ldots, N-1$. For (13), we have the following proposition.

Proposition 1: If $\hat{\mathbf{w}}_i(t) : [0, T] \to [0, 1]^{MN}$ is measurable, and $\sum_{j=1}^{M} \hat{\mathbf{w}}^j_i(t) = 1$ holds, then the function $\mathbf{w}_i(t) : [0, T] \to \{0, 1\}^{MN}$ converted from (13) using zero-order hold satisfies

$$\left\| \int_0^T \hat{\mathbf{w}}_i(\tau) - \mathbf{w}_i(\tau)\, \mathrm{d}\tau \right\|_\infty \leq \Delta t \tag{14}$$

and $\mathbf{w}_i(t)$ satisfies $\sum_{j=1}^{M} \mathbf{w}^j_i(t) = 1$.

Proposition 1 is a given as Theorem 5 in [29], which means that the approximation error of the SUR step can be made arbitrary small by choosing a sufficiently small grid length Δt. In practice, the upper bound of (14) is often conservative, leading to even smaller approximation error. In terms of computational cost, the SUR step can be neglected when compared to solving (12).

In MPC, only the first element of the solution is fed back to the system. At next sampling instant, the NLP (12) and the SUR (13) are executed again using the latest state measurement. The outline of the MPC algorithm is given in Algorithm 1.

Remark 2: The proposed MPC by design is a centralized planner. However, since each agent is independent, dynamics (7) is usually separable. As a result, problem (11) can be split into n smaller problems for each agent if no global tasks are present, e.g. global formation.

Algorithm 1 MPC Algorithm for Task Priority Planning

Input:
Number of grid points N, weighting parameters K_i, P_i.
Initialize:
Initialize $\mathbf{x}_{0|0}$
for $s = 0, 1, \ldots$ **do**
Solve problem (12) and obtain $\hat{\mathbf{w}}_{s|k}$ for $k = 0, \ldots, N-1$.
Obtain $\mathbf{w}_{s|k}$ using (13) for $k = 0, \ldots, N-1$.
Identify the index j of the non-zero element of $\mathbf{w}_{s|0}$, trigger the composite task.
end for

IV. SIMULATION

In this section, simulations are conducted where three robots perform the motion task and the obstacle avoidance task on the ground. Comparisons are made between the proposed MPC task priority planner and the traditional logic-based planner with both static and dynamic obstacles.

A. TASK DESIGN

We consider two elementary tasks, namely the motion task and the obstacle-avoidance task. They can form two composite tasks in different priority orders. Other elementary tasks, such as centroid and rigid formation [12] can also be considered but omitted here for sake of simplicity.

1) MOTION TASK

In the motion task, robots are driven toward a target point along a pre-determined reference trajectory. The corresponding task function can be defined as

$$\rho_B = p \in \mathbb{R}^2, \tag{15}$$

where $p = [p_x, p_y]^\top$ is the robot position coordinate. The desired motion task velocity can be calculated as

$$v_B = J_B^\dagger(\dot{\rho}_{B,d} + \Lambda_B \tilde{\rho}_B), \tag{16}$$

where $J_B = I_2$ is the Jacobian matrix of motion task and I_2 is the identity matrix; $J_B^\dagger$ is the pseudo-inverse of J_B. Since J_B is symmetric and idempotent, we have $J_B^\dagger = J_B$; $\rho_{B,d}$ is the desired task function; Λ_B is a suitable constant positive-definite matrix of gains; $\tilde{\rho}_B = \rho_{B,d} - \rho_B$ is the motion task error.

2) OBSTACLE-AVOIDANCE TASK

In the obstacle-avoidance task, robots must avoid obstacles detected by their sensors along the reference trajectory. The corresponding task function can be defined as

$$\rho_A = \|p - p_O\| \in \mathbb{R}, \tag{17}$$

where $p_O = [p_{xO}, p_{yO}]^\top$ is the obstacle position coordinate. The desired obstacle-avoidance task velocity can be calculated as

$$v_A = J_A^\dagger \Lambda_A \tilde{\rho}_A, \tag{18}$$

where $J_A = \hat{r}^T$, $\hat{r} = \frac{p-p_O}{\|p-p_O\|}$; $J_A^\dagger$ is the pseudo-inverse of J_A; Λ_A is a suitable constant positive-definite matrix of gains in the obstacle-avoidance task; $\tilde{\rho}_A = d - \|p - p_O\|$ is the motion task error; d is the safe distance.

TABLE 1. Parameter values used in the simulation.

Parameter	Value
obstacle avoidance task gain Λ_A	15
motion task gain Λ_B	13
safe distance d	$1m$
agent 1 initial point $\hat{x}_0$	$(0, 1)m$
agent 2 initial point $\hat{x}_0$	$(-5, 1)m$
agent 3 initial point $\hat{x}_0$	$(0, -4)m$
obstacle 1 (p_{xO}, p_{yO})	$(10, 10.5)m$
obstacle 2 (p_{xO}, p_{yO})	$(8, 12.5)m$
obstacle 3 (p_{xO}, p_{yO})	$(16, 11.5)m$
MPC sampling frequency $\frac{1}{T_s}$	20Hz
Logic-based method sampling frequency $\frac{1}{T_s}$	20Hz
MPC prediction horizon T	3s
MPC grid point number N	60

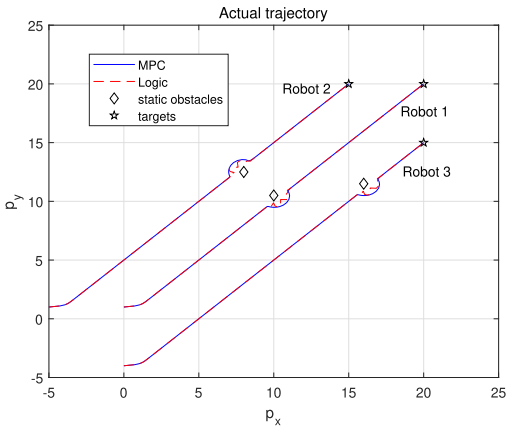


FIGURE 4. Robot trajectories of the MPC and the logic-based method when static obstacles are present.

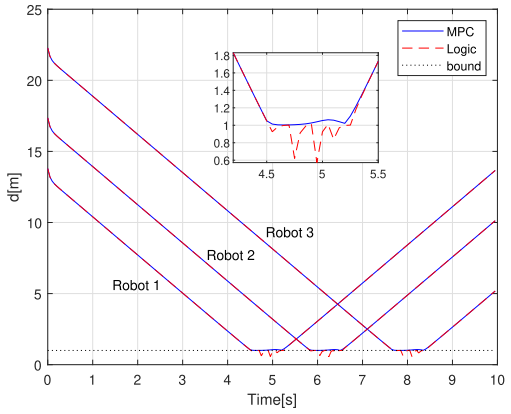


FIGURE 5. Distances between the robots and the obstacles when static obstacles are present.

3) COMPOSITION TASK

In the first composition task, the obstacle-avoidance task has higher priority, leading to the task velocity as

$$\text{Composite task A: } v_{d_1} = v_A + (I - J_A^\dagger J_A) v_B, \tag{19}$$

where $I - J_A^\dagger J_A$ is the null space of the obstacle avoidance task. In the second composition task, the motion task has higher

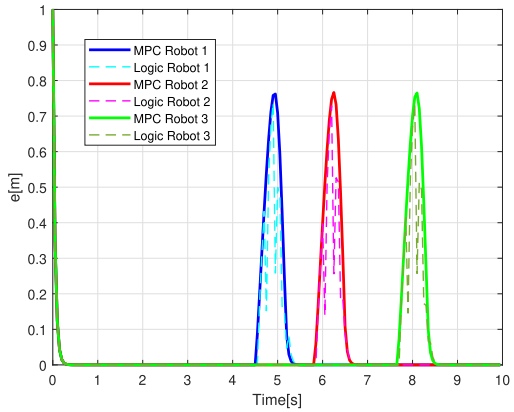


FIGURE 6. Tracking error of the robots when static obstacles are present.

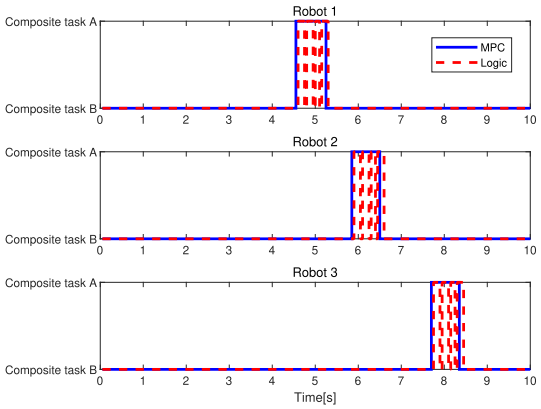


FIGURE 7. Active mode of the robots during the task execution process when static obstacles are present. Composite task A: obstacle-avoidance task>motion task; Composite task B: motion task>obstacle-avoidance task.

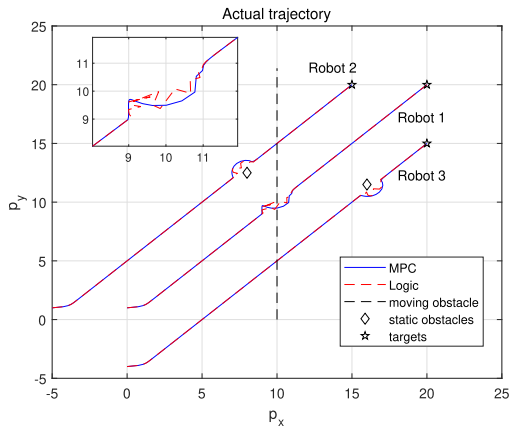


FIGURE 8. Robot trajectories of the MPC and the logic-based method when dynamic obstacles are present.

priority, leading to the task velocity as

$$\text{Composite task B: } v_{d_2} = v_B + (I - J_B^{\dagger} J_B) v_A, \tag{20}$$

where $I - J_B^{\dagger} J_B$ is the null space of the motion task.

B. SIMULATION CONFIGURATION

The objective of the motion task for the robots is to move from the initial points to the target points. The pre-defined

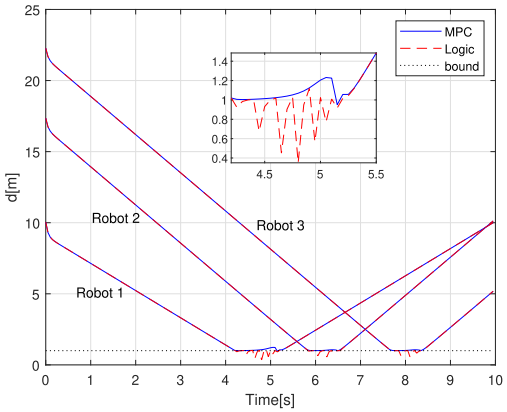


FIGURE 9. Distances between the robots and the obstacles when dynamic obstacles are present.

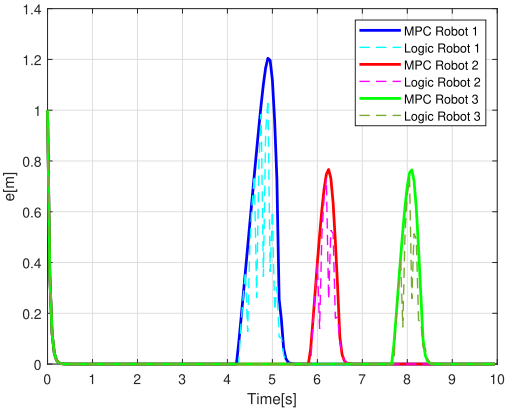


FIGURE 10. Tracking error of the robots when dynamic obstacles are present.

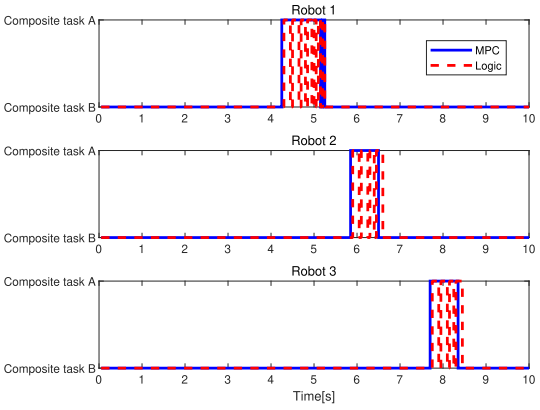
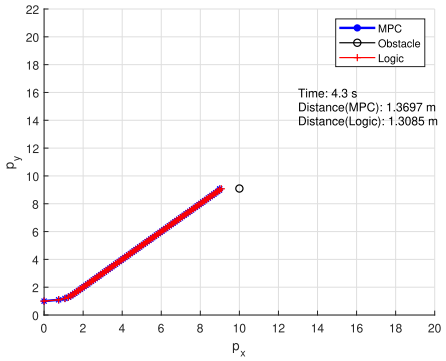


FIGURE 11. Active mode of the robots during the task execution process when dynamic obstacles are present. Composite task A: obstacle-avoidance task>motion task; Composite task B: motion task>obstacle-avoidance task.

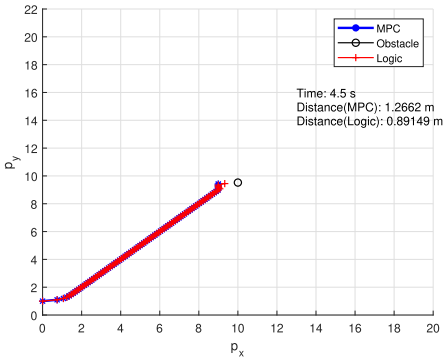
reference trajectories of the three robots are given by

$$\rho_d^1(t) = [1 + 0.9t, 1 + 0.9t], \tag{21}$$
$$\rho_d^2(t) = [-4 + 1.9t, 1 + 1.9t], \tag{22}$$
$$\rho_d^3(t) = [1 + 1.9t, -4 + 1.9t]. \tag{23}$$

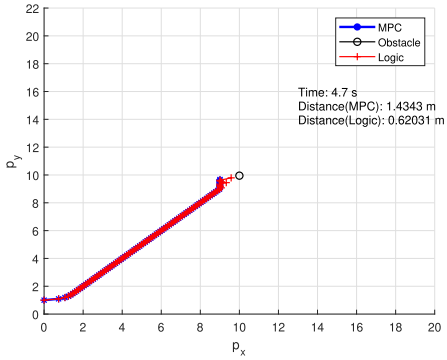
Parameters used in the simulation are shown in Table. 1. The proposed MPC algorithm is compared against a traditional



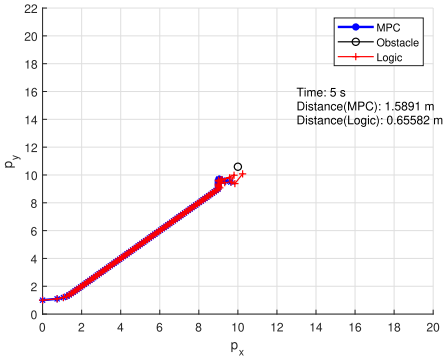
(a) At time $t = 4.3$s, robot 1 has detected an obstacle.



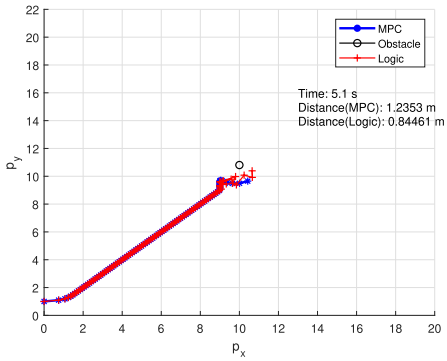
(b) At time $t = 4.5$s, the MPC planner has switched to the composite task A.



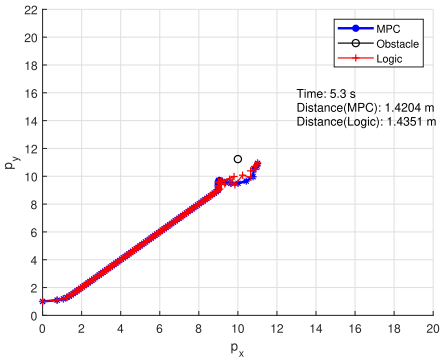
(c) At time $t = 4.7$s, the MPC planner maintains the composite task A, which produces little velocity to wait for the obstacle to pass.



(d) At time $t = 5.0$s, the MPC planner decided to switch to the composite task B.



(e) At time $t = 5.1$s, the MPC planner decided to switch back to the composite task A.



(f) At time $t = 5.3$s, the MPC planner has switched back to the composite task B and keep going.

FIGURE 12. Robot 1 trajectory using the MPC and the traditional method at different time instants when dynamic obstacles are present. Distances from robot 1 to the dynamic obstacle are shown for both methods.

task supervisor based on trivial switching logic, i.e. if the distance between the robot and the closest obstacle is smaller than the safe distance, the robot performs the composite task A, otherwise the composite task B is performed. Both methods are implemented using the same parameters in Matlab R2020a on a laptop running Windows 10 with AMD R5 4600H CPU at 3.00 Ghz and 16GB memory. The NLP (12) is solved by Ipopt [30] using Casadi toolbox [31].

In all our simulations, the MPC algorithm has an average runtime of 38 ms and a maximum of 49 ms. On the contrary, the logic-based method has a negligible execution time for just implementing a if-else based piece of code. This has demonstrated that the MPC algorithm is able to run in real-time even though in a trivial Matlab implementation. In next sub-sections, we show the proposed MPC algorithm significantly outperforms the logic-based one at the cost of such increased on-line computational burden.

C. RESULTS

1) STATIC OBSTACLES

In the first scenario, static obstacles are present at positions given in Table 1. The trajectories of the proposed MPC and the logic-based method are shown in Fig. 4. The distances between the robots and the obstacles are shown in Fig. 5. The tracking error is shown in Fig. 6.

It can be seen that the proposed MPC method significantly outperforms the logic-based method using the same sampling frequency. The trajectory of composite tasks is shown

in Fig. 7. The logic-based method switches frequently to avoid obstacles, leading to oscillations of trajectories and violations of the safe distance when encountering obstacles. On the other hand, the proposed MPC method smoothly finishes tasks using only two priority switchings, without violations of the safe distance constraint. This can be explained by the fact that the MPC method is able to predict obstacles in advance hence to prepare priority switching earlier (e.g. see Fig. 7, the MPC method always switches to composite task A earlier than the logic-based method). This demonstrates the effectiveness of the proposed MPC algorithm and its advantages over the traditional logic-based method.

2) DYNAMIC OBSTACLES

In the second scenario, we introduce a dynamic obstacle in the path of robot 1, defined by

$$(p_{xO}, p_{yO}) = (10, 2.14t)m, \tag{24}$$

meaning that the obstacle is moving in the y-direction at $p_x = 10$m with a speed of 2.14m/s. At each sampling instant, the proposed MPC method detects the position of the dynamic obstacle and makes predictions using this information without knowing the future position of the dynamic obstacle. The trajectories of the proposed MPC and the logic-based method are shown in Fig. 8. The distances between the robots and the obstacles are shown in Fig. 9. The trajectory of task priority is shown in Fig. 11. The tracking error is shown in Fig. 10. It can be observed that the logic-based method switches frequently but fails to avoid the dynamic obstacle, leading to significant violations of the safety distance constraint. The proposed MPC method only temporarily and slightly violates the safety distance constraint at around $t = 5.2$s, but quickly run away from the obstacle thereafter (see Fig. 9). This is compatible in mode switching behavior as shown in Fig. 11, where the MPC planner switches to composite task A after $t = 4.2$s but switches to composite task B at around $t = 5.1$s when it founds the obstacle is also moving in its direction. The MPC planner then switches back to composite A immediately to escape from the obstacle. This switching behavior is due to the prediction capability of MPC to avoid obstacles in advance. The slight violation of constraint is due to the introduction of slack variables in (11) to avoid frequent oscillations and switchings.

To further understand how robot 1 avoids the dynamic obstacle, Fig. 12 shows at different time points the trajectories of robot 1 using both methods. Robot 1 has detected the dynamic obstacle at around $t = 4.3$s using both methods. The proposed MPC planner immediately switches from composite task B to A while the logic-based method acts slower. At $t = 4.5$s, the MPC planner has decided to bypass the obstacle from its top (positive y direction). Then at $t = 4.7$s, the MPC planner has predicted that the safety distance constraint would be violated and re-planned the trajectory to bypass the obstacle from its bottom (negative y direction). On the other hand, the logic-based planner is too close to the obstacle at many time instants, constantly switching from two composite tasks. This example has demonstrated the prediction capability and flexibility of the proposed MPC planner in unknown and dynamic environments.

V. CONCLUSION

In the NSB control for multi-agent systems, traditional methods rely on pre-defined logic-based or fuzzy rules to dynamically plan the task priority to obtain a composite task. This paper has proposed a novel MPC-based task supervisor to plan the task priority using current and predictive information. At each sampling instant, the task priority planning problem has been formulated as a switching mode OCP, which can be solved by an efficient mixed-integer optimal control algorithm. Simulation results with static and dynamic obstacles have demonstrated the effectiveness of the proposed method in dynamic and unknown environments and its advantages over the traditional logic-based method.

REFERENCES

[1] D. Orfanus, E. P. de Freitas, and F. Eliassen, "Self-organization as a supporting paradigm for military UAV relay networks," *IEEE Commun. Lett.*, vol. 20, no. 4, pp. 804–807, Apr. 2016.
[2] Z. Ghouse, N. Hiwrale, and N. Ranjan, "Military robot for reconnaissance and surveillance using image processing," *Int. Res. J. Eng. Technol.*, vol. 4, no. 5, pp. 1767–1769, 2017.
[3] J. G. Manathara, P. B. Sujit, and R. W. Beard, "Multiple UAV coalitions for a search and prosecute mission," *J. Intell. Robotic Syst.*, vol. 62, no. 1, pp. 125–158, Apr. 2011.
[4] C. Wu, H. Fang, and X. Zeng, "Distributed object transport of mobile manipulators with optimal manipulable coordination," in *Proc. 37th Chin. Control Conf. (CCC)*, Jul. 2018, pp. 7043–7048.
[5] K. V. Najiya and M. Archana, "UAV video processing for traffic surveillence with enhanced vehicle detection," in *Proc. 2nd Int. Conf. Inventive Commun. Comput. Technol. (ICICCT)*, Apr. 2018, pp. 662–668.
[6] M. Shahvali, N. Pariz, and M. Akbariyan, "Distributed finite-time control for arbitrary switched nonlinear multi-agent systems: An observer-based approach," *Nonlinear Dyn.*, vol. 94, no. 3, pp. 2127–2142, Nov. 2018.
[7] M. Shahvali, A. Azarbahram, M.-B. Naghibi-Sistani, and J. Askari, "Bipartite consensus control for fractional-order nonlinear multi-agent systems: An output constraint approach," *Neurocomputing*, vol. 397, pp. 212–223, Jul. 2020.
[8] R. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Autom.*, vol. 2, no. 1, pp. 14–23, Mar. 1986.
[9] R. C. Arkin, "Motor schema-based mobile robot navigation," *Int. J. Robot. Res.*, vol. 8, no. 4, pp. 92–112, 1989.
[10] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Trans. Robot. Autom.*, vol. 14, no. 6, pp. 926–939, Dec. 1998.
[11] G. Antonelli and S. Chiaverini, "Kinematic control of platoons of autonomous vehicles," *IEEE Trans. Robot.*, vol. 22, no. 6, pp. 1285–1292, Dec. 2006.
[12] G. Antonelli, F. Arrichiello, and S. Chiaverini, "Experiments of formation control with multirobot systems using the null-space-based behavioral control," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 5, pp. 1173–1182, Sep. 2009.
[13] J. Chen, M. Gan, J. Huang, L. Dou, and H. Fang, "Formation control of multiple Euler–Lagrange systems via null-space-based behavioral control," *Sci. China Inf. Sci.*, vol. 59, no. 1, pp. 1–11, Jan. 2016.
[14] A. Marino, L. Parker, G. Antonelli, and F. Caccavale, "Behavioral control for multi-robot perimeter patrol: A finite state automata approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 831–836.
[15] A. Marino, L. E. Parker, G. Antonelli, and F. Caccavale, "A decentralized architecture for multi-robot systems based on the null-space-behavioral control with application to multi-robot border patrolling," *J. Intell. Robot. Syst.*, vol. 71, nos. 3–4, pp. 423–444, Sep. 2013.
[16] G. Muscio, F. Pierri, M. A. Trujillo, E. Cataldi, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero, "Coordinated control of aerial robotic manipulators: Theory and experiments," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 4, pp. 1406–1413, Jul. 2018.

[17] K. Baizid, G. Giglio, F. Pierri, M. A. Trujillo, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero, "Behavioral control of unmanned aerial vehicle manipulator systems," *Auto. Robots*, vol. 41, no. 5, pp. 1203–1220, 2017.
[18] L. Moreno, E. Moraleda, M. A. Salichs, J. R. Pimentel, and A. de la Escalera, "Fuzzy supervisor for behavioral control of autonomous systems," in *Proc. 19th Annu. Conf. IEEE Ind. Electron. (IECON)*, Nov. 1993, pp. 258–261.
[19] A. Marino, F. Caccavale, L. E. Parker, and G. Antonelli, "Fuzzy behavioral control for multi-robot border patrol," in *Proc. 17th Medit. Conf. Control Autom.*, Jun. 2009, pp. 246–251.
[20] G. Droge, P. Kingston, and M. Egerstedt, "Behavior-based switch-time MPC for mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 408–413.
[21] G. Capi, "Robot task switching in complex environments," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics*, Sep. 2007, pp. 1–6.
[22] Y. Wang, S. Li, Q. Chen, and W. Hu, "Biology inspired robot behavior selection mechanism: Using genetic algorithm," in *Proc. Int. Conf. Life Syst. Modeling Simulation*. Shanghai, China: Springer, 2007, pp. 777–786.
[23] H. Fukushima, K. Kon, and F. Matsuno, "Model predictive formation control using Branch-and-Bound compatible with collision avoidance problems," *IEEE Trans. Robot.*, vol. 29, no. 5, pp. 1308–1317, Oct. 2013.
[24] X. Qian, J. Gregoire, A. de La Fortelle, and F. Moutarde, "Decentralized model predictive control for smooth coordination of automated vehicles at intersection," in *Proc. Eur. Control Conf. (ECC)*, Jul. 2015, pp. 3452–3458.
[25] J. Huang, N. Zhou, and M. Cao, "Adaptive fuzzy behavioral control of second-order autonomous agents with prioritized missions: Theory and experiments," *IEEE Trans. Ind. Electron.*, vol. 66, no. 12, pp. 9612–9622, Dec. 2019.
[26] S. Sager, "Reformulations and algorithms for the optimization of switching decisions in nonlinear optimal control," *J. Process Control*, vol. 19, no. 8, pp. 1238–1247, Sep. 2009.
[27] J. Huang, W. Wu, Y. Ning, N. Zhou, and Z. Xu, "A behavior control scheme for multi-robot systems under human intervention," in *Proc. Chin. Control Conf. (CCC)*, Jul. 2019, pp. 6189–6193.
[28] H. G. Bock and K. J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proc. Volumes*, vol. 17, no. 2, pp. 1603–1608, Jul. 1984.
[29] S. Sager, H. G. Bock, and M. Diehl, "The integer approximation error in mixed-integer optimal control," *Math. Program.*, vol. 133, nos. 1–2, pp. 1–23, Jun. 2012.
[30] 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.
[31] 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.

YUTAO CHEN received the B.E. degree in automation from Hunan University, China, in 2012, the master's degree from the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China, in 2014, and the Ph.D. degree from the Department of Information Engineering, University of Padova, Italy, in 2018. From 2019 to 2020, he was a Postdoctoral Researcher with the Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands. He is currently an Assistant Professor with the College of Electrical Engineering and Automation, Fuzhou University, China. His research interests include model predictive control algorithms and unmanned intelligent systems and applications.

ZHENYI ZHANG (Student Member, IEEE) received the B.E. degree in mechanical and electronic engineering and the M.E. degree in mechanical engineering from Zhejiang Sci-Tech University, Hangzhou, China, in 2016 and 2019, respectively. He is currently pursuing the Ph.D. degree with Fuzhou University, China. His research interests include intelligent robot ethology and multi-agent systems.

JIE HUANG (Member, IEEE) received the B.E. degree in electrical engineering and automation and the M.E. degree in control engineering from Fuzhou University, China, in 2005 and 2010, respectively, and the Ph.D. degree in control science and engineering from the Beijing Institute of Technology, Beijing, China, in 2015. From 2005 to 2015, he was a Lecturer with the Fujian Institute of Education, Fuzhou, China. From 2014 to 2018, he held a postdoctoral position and a Lecturer with the Faculty of Science and Engineering, University of Groningen, The Netherlands. He is currently a Full Professor of robotic and control with the College of Electrical Engineering and Automation, Fuzhou University. He is the Vice-President of the Fujian Automation Association, Fujian, China. His research interests include autonomous robots, complex network dynamics, and multi-agent systems.