Distributed Model Predictive Contouring Control for Real-Time Multi-Robot Motion Planning

Xin, Jianbin; Qu, Yaoguang ; Zhang, Fangfang ; Negenborn, R.R.

DOI
10.23919/CSMS.2022.0017

Publication date
2022

Document Version
Final published version

Published in
Complex System Modeling and Simulation

COMPLEX SYSTEM MODELING AND SIMULATION
ISSN 2096-9929 01/06 pp 273−287
Volume 2, Number 4, December 2022
DOI: 10.23919/CSMS.2022.0017

Distributed Model Predictive Contouring Control for Real-Time Multi-Robot Motion Planning

Jianbin Xin, Yaoguang Qu, Fangfang Zhang*, and Rudy Negenborn

Abstract: Existing motion planning algorithms for multi-robot systems must be improved to address poor coordination and increase low real-time performance. This paper proposes a new distributed real-time motion planning method for a multi-robot system using Model Predictive Contouring Control (MPCC). MPCC allows separating the tracking accuracy and productivity, to improve productivity better than the traditional Model Predictive Control (MPC) which follows a time-dependent reference. In the proposed distributed MPCC, each robot exchanges the predicted paths of the other robots and generates the collision-free motion in a parallel manner. The proposed distributed MPCC method is tested in industrial operation scenarios in the robot simulation platform Gazebo. The simulation results show that the proposed distributed MPCC method realizes real-time multi-robot motion planning and performs better than three commonly-used planning methods (dynamic window approach, MPC, and prioritized planning).

Key words: multi-robot system; path planning; model predictive contouring control; distributed optimization

1 Introduction

Currently, robots are playing an increasingly important role in the manufacturing, defense technology, and e-commerce industries[1, 2]. In the manufacturing industry, Automated Guided Vehicles (AGVs) have been used as intelligent robots and multiple AGVs collaborate to improve operational efficiency when performing the required tasks[3, 4]. As the robot becomes more intelligent and autonomous, the AGV is being replaced by the autonomous mobile robot, which moves in a free-ranging way rather than the traditional grid roadmap used for the AGVs[5]. During the execution of these tasks, multi-robot systems, however, must have safe and collision-free motions. As a result, multi-robot motion planning has become a popular research topic in the areas of manufacturing and logistics[6].

Point-to-point planning is an important branch of multi-robot motion planning. All robots are distributed in the same workspace and required to find optimal paths connecting their respective starting and endpoints to ensure that there is no collision between each robot and the environment and no collision between every two robots at any time[1, 5]. Motion planning for multiple robots is more complex than that for a single robot. Each robot needs to consider the behavior of other robots, and the neighboring robots will become dynamic obstacles[7].

Local motion planning with dynamic obstacles and target points has been the focus of multi-robot path planning. Common local path planning methods include the artificial potential field method[8], velocity obstacle method[9], and fuzzy logic control[10]. The artificial potential field method is convenient for real-time control at the bottom level, but it easily falls into the local optimal point. The velocity obstacle method can deal with obstacles of any shape; however, the method assumes that the obstacles follow a fixed linear velocity, ignoring the bias generated by other robots

• Jianbin Xin, Yaoguang Qu, and Fangfang Zhang are with the School of Electrical and Information Engineering, Zhengzhou University, Zhengzhou 450001, China. E-mail: j.xin@zzu.edu.cn; yaoguang.qu@qq.com; zhangfangfang@zzu.edu.cn.
• Rudy Negenborn is with the Department of Marine and Transport Technology, Delft University of Technology, Delft, CD2628, the Netherlands. E-mail: r.r.negenborn@tudelft.nl.
* To whom correspondence should be addressed.
Manuscript received: 2022-05-26; revised: 2022-08-12; accepted: 2022-08-29

that may make the same collision avoidance behavior. Fuzzy logic control overcomes the potential field method's tendency to produce local optima, but the increased number of inputs can lead to problems in constructing inference rules and a dramatic expansion of the fuzzy table. These methods can quickly plan collision-free motions for the multi-robot system. Still, the robot's motion process cannot be accurately described, resulting in the inability to optimize the task performance index of multiple moving robots.

Kinematic model based multi-robot motion planning methods, which can accurately and effectively design dynamic collision avoidance strategies for obstacles and improve the performance index of the system for completing operational tasks, are currently available, such as the dynamic window method[11] and model predictive control[12, 13]. The dynamic window method is a fast planning method based on the combination of linear and angular velocities. Nevertheless, this method can only deal with static obstacles within a time window, and dynamic obstacles cannot react in advance. Model Predictive Control (MPC) is a rolling optimization method that can integrate constraints such as dynamic obstacles to effectively collaborate with collision avoidance relationships among multiple robots and handle the cooperation among multiple robots by prioritized rules[14, 15] or coordinated controllers[12, 13].

When planning motions using the MPC methods (e.g., Ref. [12]), the computational burden is large. To reduce the computation burden, distributed MPC has been investigated for coordinating the multiple-vessels[16], the multiple quadrotors[17], and formulation control of multi-agent systems[18]. However, we observe that the above distributed MPC methods are still not computationally efficient for real-time planning in a complex manufacturing or logistics working environment, in which static and dynamical obstacles must be considered together. For instance, in Ref. [17], a distributed MPC offline planning algorithm was developed to reduce the computation burden by detecting and resolving only the first collision during the planning horizon. Ferranti et al. proposed a distributed MPC algorithm based on the Alternating Direction Method of Multipliers (ADMM) to coordinate multiple vessels in Ref. [16]. The computation time relying on multiple iterations for vessel negotiations takes considerably longer than the sampling time, which is not applicable for real-time planning.

To address these limitations, this paper proposes a distributed motion planning method that combines the kinematic constraints of robots, static environment constraints, and the predicted behaviors of neighboring robots to achieve collaborative motion planning for multiple robots while ensuring real-time performance. The distributed method is based on Model Predictive Contouring Control (MPCC), which is proposed for real-time motion planning of a single mobile robot[19, 20]. The MPCC allows separating the tracking accuracy and productivity to reach a good trade-off between these two objectives. Using MPCC, the followed path is not time-dependent, and the productivity can be improved, compared with the existing MPC methods which track a time-dependent path[21]. To achieve the real-time planning, we implement a fast algorithm based on a real-time iterative scheme and automatic C-code generation. In the developed distributed MPCC, each robot exchanges its respective predicted trajectories with each other to achieve multi-robot collision-free motion planning. The developed distributed MPCC is tested in the Robot Operating System (ROS) and compared with the three commonly used planning methods (dynamic window, MPC, and prioritized planning).

The remainder of the paper is organized as follows. Section 2 introduces the multi-robot motion planning problem and the workspace used in this paper. Section 3 describes the designed distributed MPCC for real-time collision-free motion planning of multiple robots. In Section 4, simulation tests are performed by implementing the developed distributed MPCC in the ROS environment. The conclusion and future research directions are given in Section 5.

2 Problem Definition and Model Description

This section defines the multi-robot motion planning problem to be studied and then mathematically describes the robot kinematic model and related static obstacles, dynamic obstacles, and reference paths.

2.1 Problem definition

For the studied multi-robot motion planning problem, each robot moves a shared workspace. Each robot is assigned a task to move in the shortest possible time from its origin to its destination to ensure efficiency. The robots must not collide with the static environment and each other in the process of performing the task to ensure coordination between the robots.

The important assumptions are described as follows:

- Each robot can obtain its position information through the positioning system in the workplace;
- Each robot has an independent controller to execute the control algorithm;
- Each robot can exchange the information with the surrounding robots;
- Perturbation information and malfunctions do not occur during communication;
- All robots start the tasks simultaneously, and each robot is assigned a fixed starting and ending point.

2.2 Description of robot workspace

The robot's workspace $\mathcal{W}$ is a two-dimensional plane, i.e., $\mathcal{W} = \mathbb{R}^2$. The robot can move in any direction. For the research problem of robot motion planning, grid maps are widely used for environment modeling due to their simplicity and ease of implementation[22, 23]. The grid map environment is useful for motion planning; we use the occupancy grid map to model the workspace.

The core idea of the grid map is to divide the working space into several grids of the same size. All robots of the multi-robot system share the same map. Each robot occupies multiple grids and can move freely in any direction on the grid map. Each grid can be represented by three states: occupied area, free area, and unknown area. A grayscale image is used to denote the status of the grid map. Each pixel of the image represents a grid on the map. As shown in Fig. 1, the black area represents the occupied area, the white area represents the free area, and the gray area represents the unknown area.

Before introducing the detailed mathematical formulation of the robot kinematic model and the designed MPCC controller, we list all the symbols used in this paper and the related descriptions in Table 1.

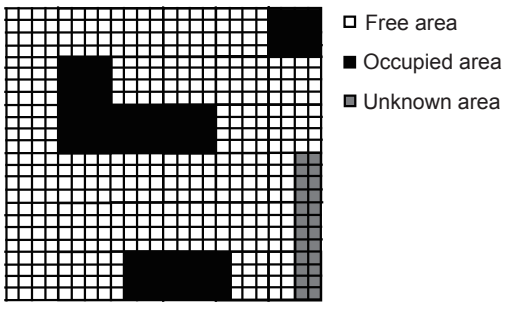


Fig. 1 Illustration of the occupancy grid map for the workspace.

Table 1 List of variables and parameters for model predictive contouring control.

Symbol		Definition
Variable	$\boldsymbol{z}_i(k)$	State of the robot i at k moment
	$\boldsymbol{u}_i(k)$	Input of robot i at k moment
	$x_i(k)$	Coordinate of robot i in the x-axis direction at k moment
	$y_i(k)$	Coordinate of robot i in the y-axis direction at k moment
	$\phi_i(k)$	Heading angle of the robot i at k moment
	$v_i(k)$	Forward speed of robot i at k moment
	$w_i(k)$	Steering speed of robot i at k moment
	$\theta_i(k)$	Distance traveled by the robot i on the reference path at k moment
Parameter	V	Total number of robots
	i	Robot i
	j	Robot j
	τ	Single time step
	N	Total number of predicted steps
	$\mathcal{W}$	Robot workspace
	$\mathcal{Z}$	Set of all feasible states
	$\mathcal{U}$	Set of all feasible inputs
	$B_i(\boldsymbol{z})$	Space occupied by robot i in state z
	$R_i^W(\boldsymbol{z})$	Rotation matrix obtained from the pose of the robot i
	c	The c-th circle of the space occupied by the robot
	$\boldsymbol{p}_i$	Position of robot i does not consider the coordinates of the z-axis
	$\boldsymbol{p}_c^i$	Coordinates of the center of the c-th circle in the body coordinate system of the robot i
	n_c	The total number of circles in the space occupied by the robot
	$\boldsymbol{q}_{0:N}^i$	Optimal trajectory of robot i
	$\boldsymbol{q}_n^i$	The n-th point of the optimal trajectory of robot i
	a	Long semi-axis of the ellipse representing the dynamic obstacle
	b	Short semi-axis of the ellipse representing the dynamic obstacle
	$R(\psi)$	Rotation matrix representing the ellipse of dynamic obstacles
	$\triangle x_{i,j}^c$	Distance between the ellipse denoting j and the c-th circle denoting i in the x-axis direction
	$\triangle y_{i,j}^c$	Distance between the ellipse denoting j and the c-th circle denoting i in the y-axis direction
	P^i	Global reference path for robot i
	$\boldsymbol{p}_{i,m}^r$	The m-th coordinate point of the reference path of robot i
	v_{ref}	Reference speed of the robot i
	$\boldsymbol{e}_i(k)$	Error vector of robot i at k moment
	$\boldsymbol{z}_{1:N}^i$	Robot i's N states on the predict trajectory
	$\boldsymbol{u}_{0:N-1}^i$	Robot i's N inputs on the predict trajectory
	$\boldsymbol{z}_{\text{init}}^i$	Initial state of robot i
	$\boldsymbol{z}_{\text{goal}}^i$	Target state of robot i

2.3 Robot kinematic model

To represent the kinematic model of each robot, for the sake of simplicity, we use the unicycle motion model. The unicycle model is used to address the displacement and velocity of two wheels of a differential drive robot, which is a common type of mobile robot[24]. To simplify the description, all robots use the same kinematic model. The kinematic model is mathematically described as follows:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\phi}\end{bmatrix}=\begin{bmatrix}\cos\phi\\ \sin\phi\\ 0\end{bmatrix}v+\begin{bmatrix}0\\ 0\\ 1\end{bmatrix}w \tag{1}$$

where x and y are the coordinates of robot i in the x-axis and y-axis directions, respectively. ϕ is the heading angle of the robot. v is the forward speed and w is the steering speed.

Due to the practical capability of the robot, we constrain the steering speed $|w|\leqslant w_{\max}$ and the forward speed $|v|\leqslant v_{\max}$ with the vehicle performance, where $w_{\max}$ is the maximum steering speed, and $v_{\max}$ is the maximum forward velocity.

The above kinematic model of robot i can be discretized using the multiple direct hitting method[25] as follows:

$$z_i(k+1)=z_i(k)+\tau\begin{bmatrix}v_i(k)\cos(\phi_i(k))\\ v_i(k)\sin(\phi_i(k))\\ \omega_i(k)\end{bmatrix},i\in I_V \tag{2}$$

where τ denotes a single time step (in seconds). We express Eq. (2) as $z_i(k+1)=f_i(z_i(k),\boldsymbol{u}_i(k))$, where $z_i(k)=[x_i(k),y_i(k),\phi_i(k)]^{\mathrm{T}}\in\mathcal{Z}$, $z_i(k)$, $x_i(k)$, $y_i(k)$, and $\phi_i(k)$ denote the state, longitudinal position, lateral position, and heading angle of robot i at moment k, respectively. $\mathcal{Z}$ denotes the set of all feasible states. $\boldsymbol{u}_i(k)=[v_i(k),\omega_i(k)]^{\mathrm{T}}\in\mathcal{U}$. $\boldsymbol{u}_i(k)$, $v_i(k)$, and $\omega_i(k)$ denote robot i's input, forward velocity, and steering velocity at the k-th moment, respectively. $\mathcal{U}$ denotes the set of all feasible inputs, $I_V:=\{1,2,...,V\}$.

The space occupied by robot i in state z is denoted as $B_i(z)$. As shown in the blue circular region in Fig. 2, n_c circles are used in this paper for an approximate representation, i.e., $B_i(z)\subseteq\bigcup_{c\in\{1,2,...,n_c\}}B_c^i(z)\subset\mathcal{W}$. The center coordinates of each circle are expressed in the inertial coordinate system as $\boldsymbol{p}_i+R_i^W(z)\boldsymbol{p}_c^i$, where $\boldsymbol{p}_i=[x_i,y_i]$ denotes the robot coordinates without considering the z-axis, $R_i^W(z)$ denotes the rotation matrix obtained from the pose of the robot, and $\boldsymbol{p}_c^i$ denotes the center coordinates of each circle in the body coordinate system.

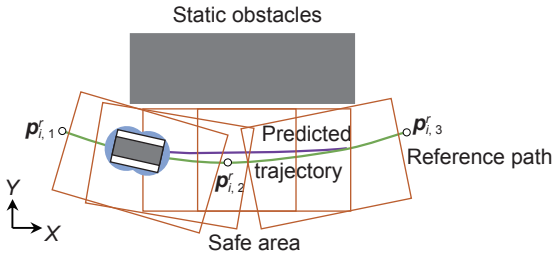


Fig. 2 Safety zones for robots in static environments.

2.4 Related constraints

This part provides the related static and dynamic obstacles and the reference path, and these contents are necessary when planning the collision-free motions based on the kinematic model.

2.4.1 Static obstacle constraints

The space occupied by the static obstacles is denoted as $O^{\text{static}}\subset\mathcal{W}$. By establishing a static map of the environment, according to the location of each robot, we can use the predicted trajectory of the last moment to calculate a set of convex quadrilaterals in the free area as a safe area for the robot, as shown in the orange box in Fig. 2.

To obtain the convex region at moment k, we first define the optimal trajectory at moment $k-1$, which is $\boldsymbol{q}_{0:N}^i=[\boldsymbol{p}_{1:N|k-1}^{i*},\boldsymbol{q}_N^i]$. It is the latest predicted trajectory at moment k, where $\boldsymbol{p}_{1:N|k-1}^{i*}=[\boldsymbol{p}_{1|k-1}^{i*},\boldsymbol{p}_{2|k-1}^{i*},...,\boldsymbol{p}_{N|k-1}^{i*}]$, $\boldsymbol{q}_N^i$ is the extrapolation of the last two points, i.e., $\boldsymbol{q}_N^i=2\boldsymbol{p}_{N|k-1}^{i*}-\boldsymbol{p}_{N-1|k-1}^{i*}$. For each point $\boldsymbol{q}_n^i(n=1,2,...,N)$, a convex region without touch is computed, denoted by four linear constraints $c_n^{\text{static}}(\boldsymbol{p}_n^i)=\bigcup_{l=1}^4 c_n^{\text{static},l}(\boldsymbol{p}_n^i)$, which makes $\boldsymbol{q}_n^i$ away from the surrounding static obstacles.

The representation of the static obstacle constraint is the same for each robot in the multi-robot system. The state of robot i at moment k is $z_i(k)$, corresponding to the constraint representation of the c-th circle that represents the space occupied by the robot and edge l of the polygon as

$$c_i^{\text{static},l,c}(z_i(k))=h^l-\boldsymbol{n}^l\cdot(\boldsymbol{p}_i-R_i^W(z)\boldsymbol{p}_c^i)>0 \tag{3}$$

where h^l and $\boldsymbol{n}^l$ denote the edges of the polygon.

2.4.2 Dynamic obstacle restraints

For the multi-robot motion planning, mutual collision avoidances need to be considered. For a particular robot, neighboring robots can be regarded as dynamic obstacles.

We use three robots, Robot 1, Robot 2, and Robot 3, to illustrate a general case, as shown in Fig. 3. Robot 2 and Robot 3 will be considered dynamic obstacles from

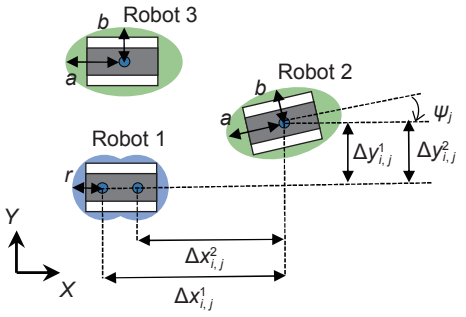


Fig. 3 Multi-robot movement schematic (from the perspective of Robot 1, dynamic obstacles are Robot 2 and Robot 3).

the perspective of Robot 1, represented by an ellipse with long semiaxis a, short semiaxis b, and a rotation matrix $R(\psi)$. At the same moment, from Robot 2's perspective, Robot 1 and Robot 3 will then be seen as dynamic obstacles. We use an ellipse to describe dynamic obstacles because the ellipse's long and short axis can be adjusted. Compared with the circle, there will not be too many redundant areas. It can better represent the moving obstacles and conform to the robot's longitudinal movement, dominating lateral movement characteristics. Moreover, the ellipse is continuous and does not produce noncontinuous derivatives at the four corners, such as a rectangle[26].

To ensure that the distance between the robots in the x-axis direction and the y-axis direction is greater than the sum of the radius of the circle indicating the space occupied by the robot and the axis length of the ellipse, at moment k, we omit the symbol k to simplify the description and express the collision avoidance constraint between the i-th robot and the other robots as follows:

$$c_{i,j}^{\text{obst, }c}(\boldsymbol{z}_i)=\begin{bmatrix}\triangle x_{i,j}^c\\ \triangle y_{i,j}^c\end{bmatrix}^{\mathrm{T}} R(\psi_j)^{\mathrm{T}}\begin{bmatrix}\frac{1}{a^2+r^2} & 0\\ 0 & \frac{1}{b^2+r^2}\end{bmatrix}R(\psi_j)\begin{bmatrix}\triangle x_{i,j}^c\\ \triangle y_{i,j}^c\end{bmatrix}>1 \tag{4}$$

where the distances between the c-th circle and the dynamic obstacle j in the x-axis direction and the y-axis direction are denoted by $\triangle x_{i,j}^c$ and $\triangle y_{i,j}^c$, respectively. r denotes the radius of the c-th circle. Taking the three robots in Fig. 3 as an example, i represents Robot 1, while j represents Robot 2 and Robot 3.

2.4.3 Reference path

Connecting the starting and ending points of the robot gives the simplest global reference path, which can also be given by the global path planner. As shown in Fig. 2, we assume that the global reference path P_i of robot i consists of a series of line segments obtained by connecting M coordinate points $\boldsymbol{p}_{i,m}^r=[x_{i,m}^r, y_{i,m}^r]\in\mathcal{W}$, where $m\in M:=\{1,2,...,M\}$. We smooth each line segment $\zeta_m^i(\theta_i)$ using a cubic polynomial, where θ_i approximates the distance traveled by robot i along the reference path.

We connect η segments of the reference path into a differentiable local reference path L_i^r for trajectory tracking.

$$L_i^r(\theta_i(k))=\sum_{h=m}^{m+\eta}\sigma_{h,+}(\theta_i(k))\sigma_{h,-}(\theta_i(k))\zeta_m^i(\theta_i(k)) \tag{5}$$

where $\sigma_{h,-}(\theta_i(k))=1/(1+\mathrm{e}^{(\theta_i-\sum_{j=m}^{h}s_h)/\epsilon})$ and $\sigma_{h,+}(\theta_i(k))=1/(1+\mathrm{e}^{(-\theta_i+\sum_{j=m}^{h-1}s_h)/\epsilon})$ are two sigmoid functions for each reference path. The function is continuous, smooth, and strictly monotonic. ϵ is a small design constant. This representation ensures that the local reference paths required to compute the solver gradient are continuous.

3 Distributed MPCC

In this section, a distributed model predictive contouring control method is developed. Model predictive contouring control is based on model predictive control[27], a control methodology to obtain control actions by minimizing an objective over a finite receding horizon based on a dynamical model. Recently, to address disturbance rejection when planning the robot motion, model predictive contouring control has been especially proposed[19, 20, 28]. The MPCC follows the desired path and adjusts the local robot motion of the complex environment in real-time, and the productivity can be improved based on the reference path. We use the foundation of the MPCC for planning the motion of a single robot[20] and further propose a distributed MPCC controller for the multi-robot planning problem. First, we introduce the mathematical formulation of the MPCC controller for each robot and then give the detailed procedures for implementing the MPCC controller for real-time planning.

3.1 Performance metrics

The performance metrics of the model prediction contour control problem in this paper are divided into $J^i(\boldsymbol{z}_i(k),\boldsymbol{u}_i(k),\theta_i(k))$ during the system state change and $J^i(\boldsymbol{z}_i(N),\theta_i(k))$ for the system end state. The goal of the model is to compute a trajectory for the robot that will not collide in the next N time steps while minimizing the cost function.

The performance metrics during the system state

change are expressed as

$$J^i(z_i(k),\boldsymbol{u}_i(k),\theta_i(k)) = J^i_{\text{tracking}}(z_i(k),\theta_i(k))+ J^i_{\text{speed}}(z_i(k),\boldsymbol{u}_i(k)) + J^i_{\text{repulsive}}(z_i(k))+ J^i_{\text{input}}(\boldsymbol{u}_i(k)) \tag{6}$$

where $J^i_{\text{tracking}}(z_i(k),\theta_i(k)), J^i_{\text{speed}}(z_i(k),\boldsymbol{u}_i(k)), J^i_{\text{repulsive}}(z_i(k))$, and $J^i_{\text{input}}(\boldsymbol{u}_i(k))$ represent the tracking cost, velocity cost, collision-free cost, and input cost, respectively, to constitute the total performance metrics of the MPCC controller for each robot, as suggested in Ref. [20].

Since the speed of the robot in the final state of the system is zero, there is no need to consider the constraints on velocity and input, so the performance metrics of the system end state are expressed as

$$J^i(z_i(N),\theta_i(N)) = J^i_{\text{tracking}}(z_i(N),\theta_i(N))+ J^i_{\text{repulsive}}(z_i(N)) \tag{7}$$

3.1.1 Tracking cost

Contour and lag errors are used to track the reference path, and we define $\bar{\boldsymbol{e}}_i(k) = \boldsymbol{p}_i(k) - \bar{\boldsymbol{p}}^r_i(\theta(k))$, then, the error vector $\boldsymbol{e}_i(k)$ is denoted as

$$\boldsymbol{e}_i(k) = \begin{bmatrix} \sin\phi(\theta_i(k)) & -\cos\phi(\theta_i(k)) \\ -\cos\phi(\theta_i(k)) & -\sin\phi(\theta_i(k)) \end{bmatrix} \bar{e}_i(k) \tag{8}$$

where $\phi(\theta_i(k)) = \arctan(\partial y^r_i(\theta_i(k))/\partial x^r_i(\theta_i(k)))$ represents the direction of the motion.

The tracking cost of model predictive contouring control is expressed as

$$J^i_{\text{tracking}}(z_i(k),\theta_i(k)) = \boldsymbol{e}_i(k)^{\mathrm{T}} Q_\epsilon \boldsymbol{e}_i(k) \tag{9}$$

where Q_ϵ is a design parameter. The solution for minimizing the quadratic tracking cost defined in Eq. (9) drives the robot toward the reference path.

3.1.2 Velocity cost

We introduce a velocity cost term to penalize the deviation of robot i's velocity $v_i(k)$ at moment k from the reference velocity v_{ref}, denoted as

$$J^i_{\text{speed}}(z_i(k),\boldsymbol{u}_i(k)) = Q_v(v_{\text{ref}} - v_i(k))^2 \tag{10}$$

where Q_v is a design parameter, the reference velocity v_{ref} can be obtained by upper-level planning, and different reference velocities can be chosen for different local reference paths.

3.1.3 Collision-free cost

We also add a cost term similar to the potential function to increase the safe distance between the robots, denoted as

$$J^i_{\text{repulsive}} z_i(k)) = Q_R \sum_{j=1}^{n} \frac{1}{(\triangle x_{i,j}(k))^2 + (\triangle y_{i,j}(k))^2 + \gamma} \tag{11}$$

where Q_R is a design parameter, n denotes the number of dynamic obstacles, $\triangle x_{i,j}(k)$ and $\triangle y_{i,j}(k)$ represent the distances between the robot and the dynamic obstacles in the x-axis direction and the y-axis direction, respectively, $\gamma \geqslant 0$ is to ensure numerical stability, and Eq. (11) also increases the robustness of the method to localization uncertainty by increasing the distance relative to the obstacles.

3.1.4 Input cost

In the end, the input cost also needs to be included as follows:

$$J^i_{\text{input}}(z_i(k),\theta_i(k)) = \boldsymbol{u}_i(k)^{\mathrm{T}} Q_u \boldsymbol{u}_i(k) \tag{12}$$

where Q_u is a design parameter.

3.2 Overall formulation

After providing the performance metrics of the MPCC controller for each robot, we give the overall mathematical formulation of the distributed MPCC control problem, which is as follows:

$$J^*_i = \min_{z^i_{0:N}, \boldsymbol{u}^i_{0:N-1}, \theta^i_{0:N}} \sum_{t=0}^{N-1} J^i(z_i(k+t),\boldsymbol{u}_i(k+t),\theta_i(k+t))+ J^i(z_i(k+N),\theta_i(k+N)) \tag{13}$$

s.t.

$$z_i(k+1) = f(z_i(k),\boldsymbol{u}_i(k)) \tag{14}$$

$$\theta_i(k+1) = \theta_i(k) + v_i(k)\tau \tag{15}$$

$$\boldsymbol{u}_i(k) \in \mathcal{U}, z_i(k) \in \mathcal{Z}, z_i(0), \theta_i(0) \text{ given} \tag{16}$$

$$c^{\text{static},l,c}_i(z_i(k)) > 0, \forall c \in \{1,2,...,n_c\}, l \in \{1,2,...,4\} \tag{17}$$

$$c^{\text{obst},c}_{i,j}(z_i(k)) > 1, \forall c \in \{1,2,...,n_c\}, \forall j \tag{18}$$

where $v_i(k)$ denotes the forward velocity of robot i, τ denotes the time step, and $\mathcal{U}$ and $\mathcal{Z}$ denote the allowed inputs and states, respectively. $z^i_{1:N}$ and $\boldsymbol{u}^i_{0:N-1}$ denote the N states and control inputs on the predictive horizon, respectively. $\theta_i(k)$ denotes the predicted travel distance of the k-th moment along the reference path. By solving this optimization problem, the locally optimal control command $[\boldsymbol{u}_i(k)^*]^{N-1}_{k=0}$ that guides the robot along the reference path without collisions is obtained.

3.2.1 Distributed architecture

Compared with a single controller to centrally plan the motions of all robots, this paper proposed a distributed control mechanism of sending and receiving other

robots' predicted trajectory between robots to ensure that each robot can solve its optimization problems in parallel through independent controllers.

The distributed network architecture of the multi-robot system is shown in Fig. 4. At the k-th moment, each robot simultaneously solves its own model predictive contouring control problem in parallel based on the predicted trajectory information of the neighboring robots at the $(k-1)$-th moment and exchanges the recalculated predicted trajectory with the neighboring robots at the current moment until it reaches its respective target point. Note that the communication between two robots may be limited in the industrial workplace. The dashed line indicates Robots 1 and V can communicate only within a limited area when they are close to each other.

The procedures of the developed MPCC method are described in Algorithm 1. Each robot uses its planner to solve the optimization problem in a parallel manner.

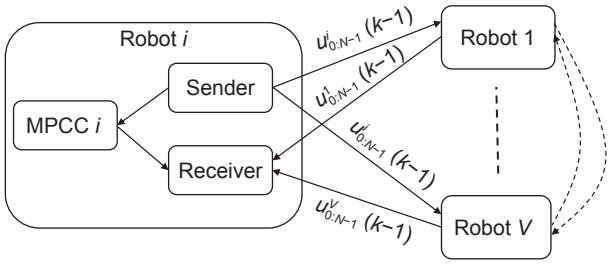


Fig. 4 Multi-robot system architecture.

Algorithm 1 Proposed distributed MPCC algorithm for multi-robot motion planning

1:Initializing parameters of all robots: $z^1_{\text{init}}, z^2_{\text{init}}, ..., z^V_{\text{init}}$, $z^1_{\text{goal}}, z^2_{\text{goal}}, ..., z^V_{\text{goal}}$ and predicted steps N
2: **for** $k = 0, t, 2t, ...$ **do**
3: for all Robot i, $i \in V$ **do in parallel**
4: Estimate the distance $\theta_i(0)$ that robot i is currently traveling on the reference path
5: The given reference path is serialized to obtain $\overline{\boldsymbol{p}}^r_i(\theta_i(k)), k = 1, 2, ..., N$, according to Eq. (5)
6: Calculate the static obstacle avoidance constraint $c^{\text{static},l,c}_i(\boldsymbol{z}_i(k))$, according to Eq. (3)
7: Receive the trajectory of the surrounding robots at $k-1$ moment
8: Add the trajectory of the surrounding robots to dynamic collision avoidance constraint $c^{\text{obst},\,c}_{i,j}(\boldsymbol{z}_i)$, according to Eq. (4)
9: Solve the optimization problem of Eq. (13)
10: Send the predicted trajectory of the current moment to other robots
11: Apply $\boldsymbol{u}^*_i(0)$ to move forward
12: **end for**
13: **end for**

First, each robot calculates the static constraint based on the reference path and static environment information (Step 6). The predictive trajectory of robot i is illustrated in Fig. 5, and the predictive region (area of circles) is occupied only by robot i to prevent the collisions from the other robots. If no feasible solution is detected, we let $v_{\text{ref}} = 0$ to stop the robot, as shown in Fig. 6. The robots then exchange information about each other's predicted trajectories (Step 7). Neighboring robots are considered dynamic obstacles added to the dynamic collision avoidance constraint (Step 8). The optimal trajectory for robot i can be obtained by solving an individual problem (Step 9). Finally, the new information is sent to the surrounding robots, while the first step of the control inputs is executed (Steps 10 and 11).

3.2.2 Real-time implementation

We use an open-source Automatic Control and Dynamic Optimization (ACADO)[29] code generation toolkit to implement the proposed distributed MPCC controller. The toolkit generates an optimized and independent C-code that restricts the relevant computations to the most basic steps. The generated C-code is based on a multiple direct hitting method and a Gauss-Legerdale 4th-order integrator implementation with a sampling time of 100 ms. The toolkit uses sequential quadratic programming to solve the nonlinear model predictive control problem of Eq. (13), qpOASES[30] is used to solve the corresponding quadratic programming problem and KKT is set to 10^{-4}. The maximum number of iterations is 10.

4 Results and Discussions

To verify the effectiveness of the proposed control

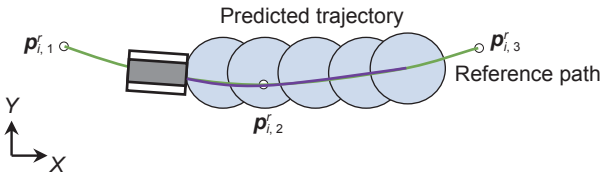


Fig. 5 Predicted trajectory based on MPCC.

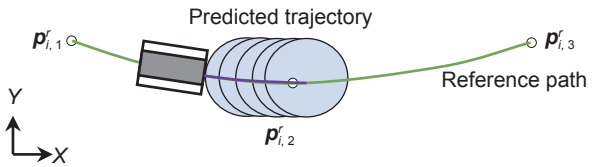


Fig. 6 Predicted trajectory if no feasible solution is detected (we let $v_{\text{ref}} = 0$).

methodology, in this section, we design two sets of scenarios while considering the method of handling collision avoidance among robots. We simulate and compare the proposed MPCC with the classical dynamic window method[11] and prioritized planning method[14, 15].

4.1 Simulation settings

We use Gazebo to build the physical simulation environment. The simulation experiment uses a computer with a 2.60 GHz main frequency and 12.0 GB memory. The robots in the experiments use the same URDF model to describe the robot's shape, size, and physical properties, all of which are jackal robots from ClearPath robotics. We choose $w_{\max}$ = 1 rad/s, $v_{\max}$ = 2 m/s, $a_{\min}$ = 0, and $a_{\max}$ = 2.5 m/s^2.

The individual functional modules of the robot were written in the C++ programming language and placed in their own separate ROS packages depending on their functions. Robot localization is implemented with the help of the amcl package, which implements the adaptive Monte Carlo localization method[31]. The method is based on real-time data obtained from radar and uses particle filters to track the pose of the robot based on the known map information. In addition, a map server node is used to provide the robot with map information of its surroundings, and the RViz visualization tool is used to monitor the operational state of the whole multi-robot system.

The proposed MPCC method is compared with three commonly-used methods:

● Dynamical Window Approach (DWA), which is an effective online collision-free local planner for mobile robots[11]. DWA calculates the search space from a set of velocities that produce a safe trajectory, then selects the optimal velocity and heading to maximize the robot's clearance.

● Prioritized planning method, in which the motion of each robot is planned sequentially following different priorities[9]. In this paper, the prioritized planning strategy is integrated with the MPCC (referred to as P-MPCC) for re-planning at each sample instant.

● Model predictive control method, which uses a nonlinear predictive control for online motion planning with a time-dependent reference[21]. Here, the MPC method is also implemented in the same distributed way (D-MPC) as the proposed D-MPCC method.

The reference paths of these four methods are identical. The reference path is determined by the global planning algorithm, such as Dijkstra.

We designed a single-channel scenario of size 16 m× 6 m, using two robots for testing. The second scenario is a complex working scenario[32] of size 16 m × 11 m, tested with six robots simultaneously. For each scenario, ten simulations have been conducted.

4.2 Single channel scenario

To evaluate the multi-robot distributed MPCC method proposed in this paper, we conducted ten sets of test experiments in a single-channel scenario, as shown in Fig. 7, for different reference velocities $v_{\text{ref}} \in \{0.8\ \text{m/s}, 1.0\ \text{m/s}, 1.2\ \text{m/s}\}$ and different predictive horizons $N \in \{10, 20, 30, 40, 50\}$ ($T_{\text{Horizon}} \in \{1\ \text{s}, 2\ \text{s}, 3\ \text{s}, 4\ \text{s}, 5\ \text{s}\}$), respectively. In this scenario, the coordinates of the starting position and the coordinates of the ending position of the two robots are given in Table 2.

We evaluate the total time and distance traveled by each robot to complete their tasks, and these indices are used for comparative analysis to select the proper reference velocity v_{ref} and prediction horizon N of the MPCC controller. The total travel time, the total travel distance, and the maximum computation time under different parameter configurations are presented in Figs. 8, 9, and 10, respectively.

Figure 8 shows that the total travel time for the robots to complete their tasks fluctuates when varying N for the same reference velocity v_{ref}. The minimum of the total travel time is obtained when $N = 20$ ($T_{\text{Horizon}} = 2$ s) for each reference velocity. The configuration that $v_{\text{ref}} = 1.2$ m/s and $N = 20$ achieves the shortest travel time. When the predictive horizon is too short, the robot cannot accurately calculate the surrounding collision-free area based on the

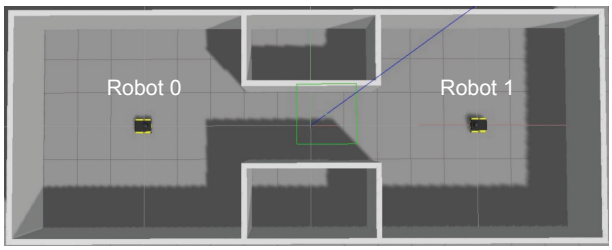


Fig. 7 Single channel scenario built with Gazebo.

Table 2 Start point and end point coordinates of the robot.

Robot	Starting point	End point
Robot 0	(−5.0, 0.0)	(5.0, 0.0)
Robot 1	(5.0, 0.0)	(−5.0, 0.0)

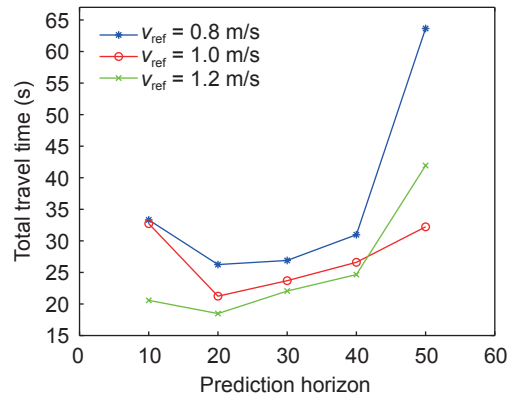


Fig. 8 Comparison of the total travel time under different v_{ref} and N.

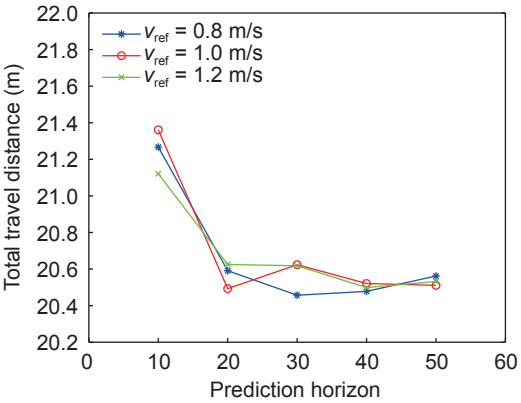


Fig. 9 Comparison of the total travel distance under different v_{ref} and N.

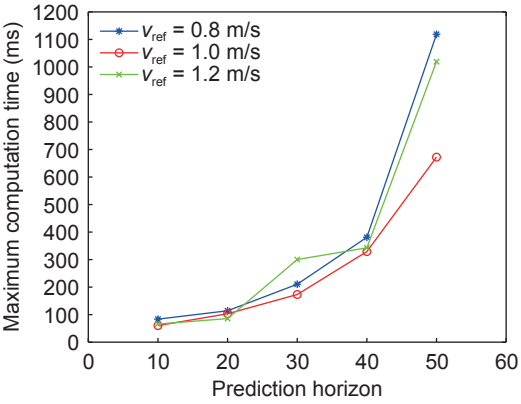


Fig. 10 Maximal computation time under different v_{ref} and N.

information of the surrounding static environment, which limits the robot's speed. When a long prediction horizon is considered, the predicted collision-free area occupied by each robot becomes large. Because each robot uses the predicted trajectories of other robots in the previous moment to perform collision avoidance, the robots may have to decelerate or wait unnecessarily for avoiding collision in advance, leading to an increase in the total travel time.

Figure 9 compares the total travel distance of the robots considering different v_{ref} and N. It is observed that the travel distance decreases and levels off when the prediction horizon N increases. The travel distance fluctuates slightly when N grows from 20 to 50.

Figure 10 records the maximum computation time under different reference velocities and the prediction horizons. Figure 10 indicates that the computation time grows when the prediction horizon N increases because the size of the optimization problem rises. For real-time planning, N is suggested to be selected at a small value of prediction horizons.

Based on the results above, in the following simulations, we choose the configuration of $v_{ref} = 1.2$ m/s and $N = 20$, since this combination provides the minimal travel time at a short computational time. For MPCC, if N is selected sufficiently long, the stability and feasibility can be ensured, as discussed in Ref. [33]. However, we focus on real-time planning, the computation burden must be reduced and a sufficiently long prediction horizon cannot be selected. The prediction horizon $N = 20$ is carefully selected, such that the travel distance is close to the one when a long planning horizon is considered.

4.3 Complex operation scenario

The second scenario is a complex industrial environment (see Fig. 11) containing both multiple static obstacles in the environment and dynamic obstacles represented by the moving robots, as proposed in Ref. [32]. We use this scenario to test the scalability of the proposed D-MPCC and the real-time

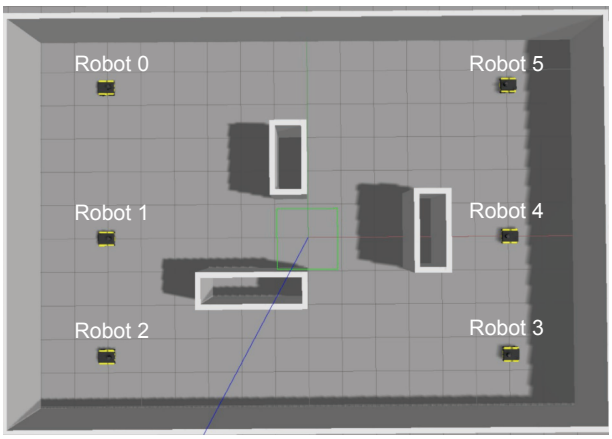


Fig. 11 Complex working scenarios built with Gazebo.

planning capability in such a complex environment. In this scenario, six mobile robots move from their pickup points to their target points. We establish the coordinate system with the center point of this scenario as the coordinate origin; each robot's starting and ending coordinates are shown in Table 3.

4.3.1 General performance

Table 4 compares the travel time and distance of each robot to complete their transport tasks of the four methods (DWA, D-MPC, P-MPCC, and D-MPCC). In the manufacturing and logistics environment, completion time is more crucial than the distance for the operator, because completion time is directly related to productivity. It can be seen from Table 4 that the proposed D-MPCC method achieves the shortest travel time for each robot, and in total their tasks are all completed as soon as possible (67.624 s). The performances of D-MPC and P-MPCC are quite close to each other. For each robot, D-MPC and P-MPCC both obtain a shorter travel time than the DWA method. Because the MPCC planner includes the reference velocity cost in the objective function while the DWA method does evaluate the reference velocity explicitly, the MPCC methods have higher productivity than the DWA method.

It can be seen from Table 4 that the proposed D-MPCC method is competitive among the four methods concerning the total travel distance. Although the D-MPCC method does not have the shortest travel distance in total, the gap is very small between the D-MPCC method (75.941 m) and the DWA method which owns the shortest total travel distance (75.248 m). Regarding the travel distance for each robot, none of these methods dominates the others.

Table 3 Coordinates of the start and end points of each robot.

Robot	Start point	End point
Robot 0	(−6.0, 4.5)	(5.0, −3.5)
Robot 1	(−6.0, 0.0)	(5.0, 4.5)
Robot 2	(−6.0, −3.5)	(5.0, 0.0)
Robot 3	(6.0, −3.5)	(−5.0, −3.5)
Robot 4	(6.0, 0.0)	(−5.0, 0.0)
Robot 5	(6.0, 4.5)	(−5.0, 4.5)

4.3.2 Trajectory analysis

In this part, we further analyze the robot trajectories determined by the proposed D-MPCC method, in comparison to DWA, D-MPC, and P-MPCC. The robot trajectories in the simulation are presented in Figs. 12–15. Their velocity evolutions for each robot are given in Figs. 16–19. In all the simulation tests, no collisions take place using these three methods.

Figures 12 and 13 give the robot trajectories computed by DWA and D-MPC, respectively. Compared to the MPCC methods in Figs. 14 and 15, the trajectory dots of DWA and D-MPC are less intensive because the DWA method may compute the trajectory in a longer computation time. As the result, the trajectories of these robots cannot be adjusted at a high frequency to possibly increase the travel time. Since the DWA methods cannot incorporate the reference velocity into the trajectory evaluation, all the robots accelerate and decelerate several times, resulting in a long travel time for each robot. The D-MPC method tracks a time-dependent reference for each robot, and the robot trajectories of D-MPC are not as smooth as the MPCC methods.

Figure 14 presents the motion process of the six robots obtained by the P-MPCC method. Note that priority increases from Robot 0 to Robot 5. When two robots meet, the high-priority robot plans its motion without considering the low-priority robot. Then the low-priority robot unilaterally avoids the high-priority robot taking its planned motion into account. This explains why Robot 5's motion is smoother than Robot 0's when the two robots meet, and this also

Table 4 Comparison of travel time and distance using different methods.

Robot	DWA		D-MPC		P-MPCC		Proposed D-MPCC	
	Travel time (s)	Travel distance (m)	Travel time (s)	Travel distance (m)	Travel time (s)	Travel distance (m)	Travel time (s)	Travel distance (m)
Robot 0	36.205±0.375	13.870±0.092	15.712±0.098	15.972±0.101	16.135±0.030	15.431±0.257	**13.097±0.111**	15.389±0.026
Robot 1	22.309±0.091	12.646±0.229	11.828±0.067	13.258±0.023	11.887±0.252	14.007±0.155	**11.332±0.034**	13.195±0.079
Robot 2	23.203±0.127	11.898±0.025	11.800±0.048	12.081±0.019	11.782±0.083	13.391±0.043	**11.441±0.071**	12.351±0.034
Robot 3	17.611±0.221	10.697±0.048	10.513±0.015	11.878±0.092	10.291±0.078	11.209±0.067	**10.251±0.026**	11.227±0.053
Robot 4	54.101±0.075	15.356±0.257	12.608±0.071	12.632±0.065	11.789±0.011	12.390±0.046	**11.752±0.151**	12.483±0.037
Robot 5	17.501±0.014	10.781±0.093	10.405±0.087	11.881±0.031	10.372±0.037	11.109±0.035	**9.751±0.173**	11.296±0.083
Sum	170.930±0.051	75.248±0.078	72.866±0.053	77.702±0.028	72.256±0.049	77.537±0.201	**67.624±0.027**	75.941±0.045

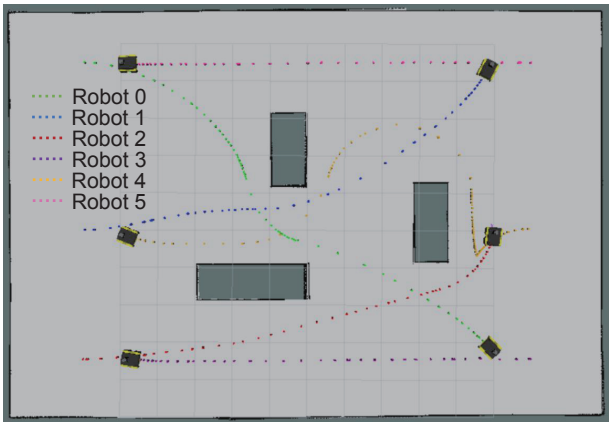


Fig. 12 Planned robot trajectories obtained using the DWA method.

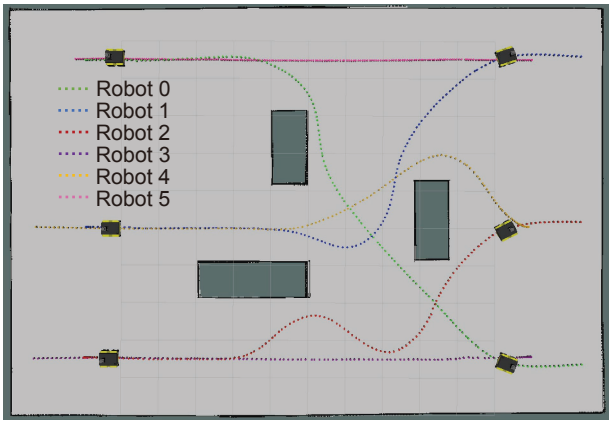


Fig. 14 Planned robot trajectories obtained using the P-MPCC method.

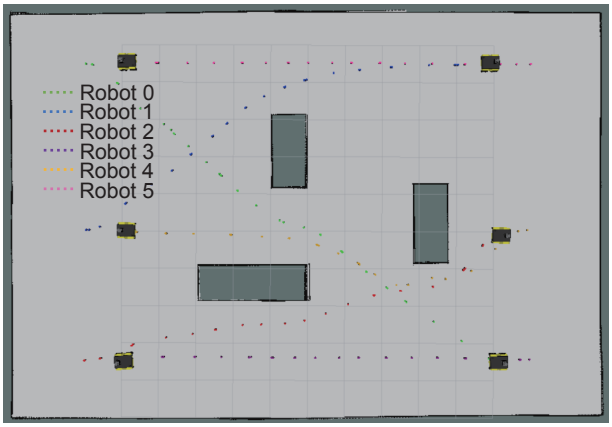


Fig. 13 Planned robot trajectories obtained using the D-MPC method.

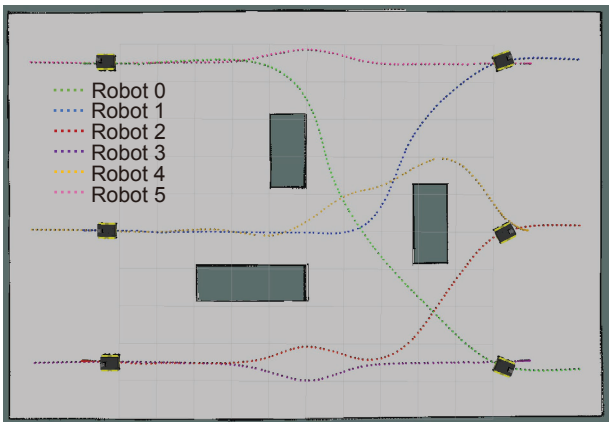


Fig. 15 Planned robot trajectories obtained using the proposed D-MPCC method.

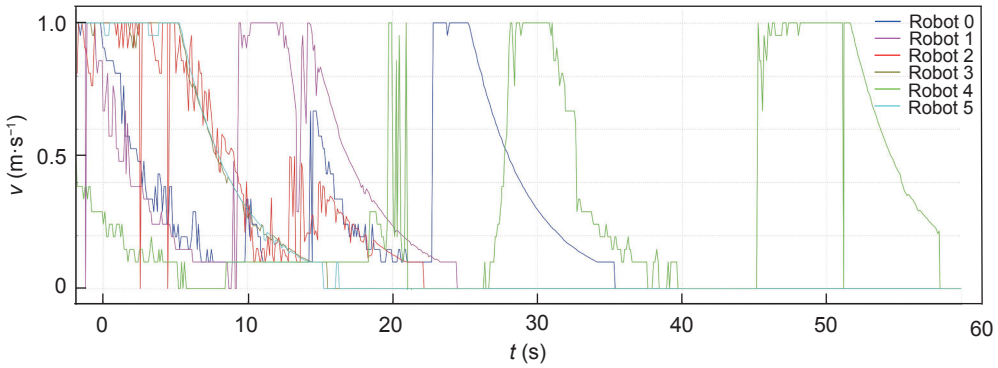


Fig. 16 Velocity evolution by using the DWA method.

applies to the other robot comparisons. When the high-priority robots move faster, the low-priority robots may not escape in time and cause a collision.

Figure 15 records the trajectories of the six robots determined by the proposed D-MPCC method. It can be seen that all the robots show cooperative behavior when meeting each other in a constrained environment. Considering the predicted trajectory of the other robot, each robot avoids collision with the other robot mutually. As a result, the trajectory curvatures are shared by the two robots to reduce their total travel time. The robot could reduce the numbers of accelerations/decelerations to achieve a shorter travel time, as illustrated in Fig. 19.

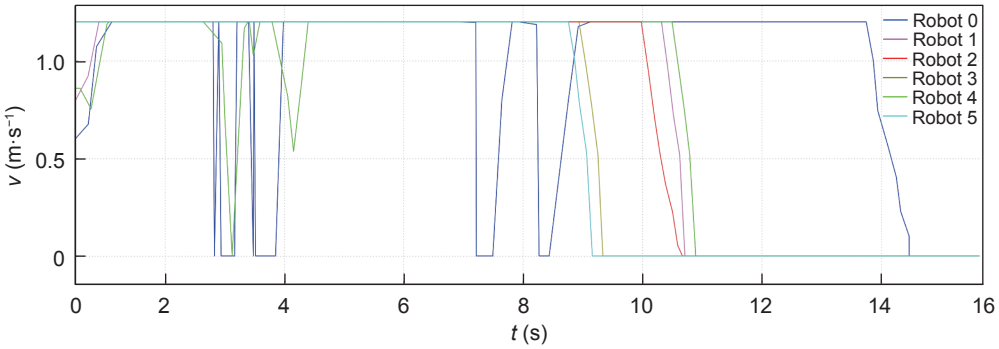


Fig. 17 Velocity evolution by using the D-MPC method.

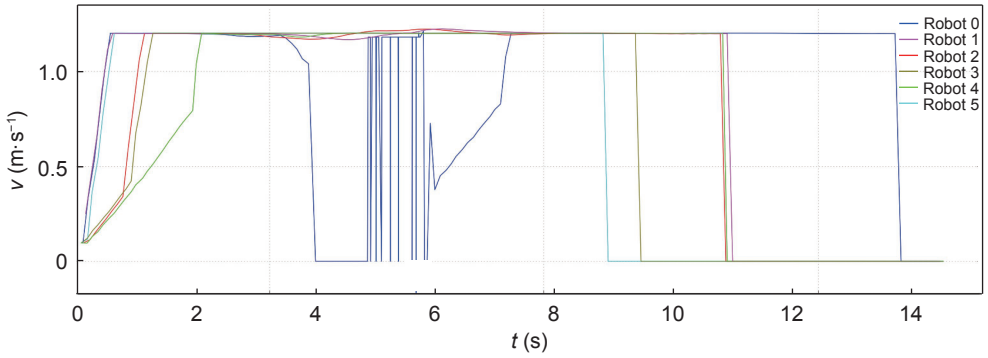


Fig. 18 Velocity evolution by using the P-MPCC method.

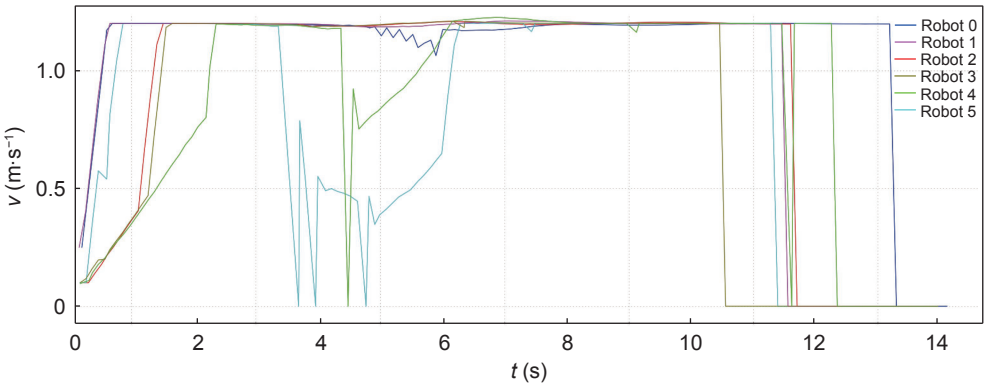


Fig. 19 Velocity evolution by using the proposed D-MPCC method.

Comparing the velocity evolutions of the four methods given in Figs. 16–19, the velocity of the DWA method for each robot requires more multiple adjustments than the MPCC methods in the constrained environment. This explains in general the DWA method has a longer travel time than the MPCC methods which explicitly track the reference velocity when determining the robot trajectory.

Figures 16–19 also show that the proposed D-MPCC method results in a more averaged travel time for each robot than the DWA, D-MPC, and P-MPCC methods. The gap in the travel time between all the robots is smaller than the other methods. These results indicate the cooperative behaviors resulting from the distributed MPCC method have a positive influence on improving the productivity of the entire robot group.

5 Conclusion and Future Research

This paper proposes a distributed Model Predictive Contour Control (MPCC) methodology for planning collision-free motions of multiple mobile robots to improve coordination and real-time performance. With the help of constraint optimization, the MPCC method fully considers the robot's kinematics constraints, static environmental constraints, and dynamic obstacle

avoidance constraints. The proposed distributed MPCC method follows the desired path and adjusts the local motion of the complex environment in real-time to improve productivity when completing the assigned tasks. Real-time coordination among robots is achieved by passing their predicted trajectories from the last moment to the surrounding robots in parallel and incorporating the predicted trajectory of the surrounding robots into the collision avoidance constraints for handling dynamic obstacles. The distributed structure improves the real-time performance of each robot to quickly solve its motion planning and ensures the scalability of the entire system.

The method proposed in this paper is validated in the complex industrial environment of the robot simulation software Gazebo. Compared with the methods of DWA, MPC, and prioritized planning, the proposed MPCC adopts a distributed structure where the predicted path information of the previous moment is exchanged between robots and added to the respective dynamic obstacle avoidance constraints. The initiative of each robot to complete the collision-avoidance action can better deal with the collision avoidance problem between multiple robots and reduce the working time and distance. The predicted trajectory is reserved for each robot to avoid the collision. If the predictive horizon is too small, the safe area of the robot will expand in a small area, limiting the robot's behavior. If the predictive horizon is too large, the robot will take unnecessary obstacle avoidance actions in advance.

When implementing the proposed MPCC method, the maximum number of robots is limited to 10, due to the limited number of channels to synchronize when using the message filters package in ROS. In future research, we will investigate a more scalable method to implement the MPCC method. Future research will also consider validation in real application scenarios and the inclusion of functions such as task assignment and global path planning.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 62173311, 61703372, and 61603345), and in part by the College Youth Backbone Teacher Project of Henan Province (No. 2021GGJS001), Henan Scientific and Technological Research Project (Nos. 222102220123 and 212102310050), the Training Project of Zhengzhou University (No. JC21640030), and the China Postdoctoral Science Foundation (No. 2020M682346).

References

[1] B. Xin, J. Zhang, J. Chen, Q. Wang, and Y. Qu, Overview of research on transformation of multi-AUV formations, *Complex System Modeling and Simulation*, vol. 1, no. 1, pp. 1–14, 2021.

[2] N. Zhao, G. Lodewijks, Z. Fu, Y. Sun, and Y. Sun, Trajectory predictions with details in a robotic twin-crane system, *Complex System Modeling and Simulation*, vol. 2, no. 1, pp. 1–17, 2022.

[3] T. Nishi, S. Akiyama, T. Higashi, and K. Kumagai, Cell-based local search heuristics for guide path design of automated guided vehicle systems with dynamic multicommodity flow, *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 2, pp. 966–980, 2019.

[4] J. Xin, L. Wei, D. Wang, and H. Xuan, Receding horizon path planning of automated guided vehicles using a time-space network model, *Optimal Control Applications and Methods*, vol. 41, no. 6, pp. 1889–1903, 2020.

[5] M. D. Ryck, M. Versteyhe, and F. Debrouwere, Automated guided vehicle systems, state-of-the-art control algorithms and techniques, *Journal of Manufacturing Systems*, vol. 54, pp. 152–173, 2020.

[6] D. Wang, H. Wang, and L. Liu, Unknown environment exploration of multi-robot system with the FORDPSO, *Swarm and Evolutionary Computation*, vol. 26, pp. 157–174, 2016.

[7] W. Schwarting, J. Alonso-Mora, and D. Rus, Planning and decision-making for autonomous vehicles, *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 187–210, 2018.

[8] N. Malone, H. T. Chiang, K. Lesser, M. Oishi, and L. Tapia, Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field, *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1124–1138, 2017.

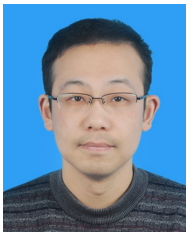
[9] J. V. D. Berg, M. Lin, and D. Manocha, Reciprocal velocity obstacles for real-time multi-agent navigation, in *Proc. 2008 IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, 2008, pp. 1928–1935.

[10] O. Castillo, H. Neyoy, J. Soria, P. Melin, and F. Valdez, A new approach for dynamic fuzzy logic parameter tuning in ant colony optimization and its application in fuzzy control of a mobile robot, *Applied Soft Computing*, vol. 28, pp. 150–159, 2015.

[11] D. Fox, W. Burgard, and S. Thrun, The dynamic window approach to collision avoidance, *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[12] Y. Kuwata and J. P. How, Cooperative distributed robust trajectory optimization using receding horizon MILP,

IEEE Transactions on Control Systems Technology, vol. 19, no. 2, pp. 423–431, 2010.
[13] H. Zheng, R. R. Negenborn, and G. Lodewijks, Fast ADMM for distributed model predictive control of cooperative waterborne AGVs, *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1406–1413, 2016.
[14] M. Čáp, P. Novák, A. Kleiner, and M. Selecký, Prioritized planning algorithms for trajectory coordination of multiple mobile robots, *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 835–849, 2015.
[15] G. Demesure, M. Defoort, A. Bekrar, D. Trentesaux, and M. Djemai, Decentralized motion planning and scheduling of AGVs in an FMS, *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1744–1752, 2017.
[16] L. Ferranti, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, Coordination of multiple vessels via distributed nonlinear model predictive control, in *Proc. 2018 European Control Conference* (*ECC*), Limassol, Cyprus, 2018, pp. 2523–2528.
[17] C. E. Luis and A. P. Schoellig, Trajectory generation for multiagent point-to-point transitions via distributed model predictive control, *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 375–382, 2019.
[18] Y. Wang, Y. Yang, Y. Pu, and C. Manzie, Path following by formations of agents with collision avoidance guarantees using distributed model predictive control, in *Proc. 2021 American Control Conference* (*ACC*), New Orleans, LA, USA, 2021, pp. 3352–3357.
[19] D. Lam, C. Manzie, and M. C. Good, Model predictive contouring control for biaxial systems, *IEEE Transactions on Control Systems Technology*, vol. 21, no. 2, pp. 552–559, 2012.
[20] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, Model predictive contouring control for collision avoidance in unstructured dynamic environments, *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, 2019.
[21] C. Rösmann, A. Makarow, and T. Bertram, Online motion planning based on nonlinear model predictive control with non-Euclidean rotation groups, in *Proc. 2021 European Control Conference* (*ECC*), Delft, the Netherlands, 2021, pp. 1583–1590.
[22] E. G. Tsardoulias, A. Iliakopoulou, A. Kargakos, and L. Petrou, A review of global path planning methods for occupancy grid maps regardless of obstacle density, *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1, pp. 829–858, 2016.
[23] H. Lu, S. Yang, M. Zhao, and S. Cheng, Multi-robot indoor environment map building based on multi-stage optimization method, *Complex System Modeling and Simulation*, vol. 1, no. 2, pp. 145–161, 2021.
[24] J. Choi, G. Lee, and C. Lee, Reinforcement learning-based dynamic obstacle avoidance and integration of path planning, *Intelligent Service Robotics*, vol. 14, no. 5, pp. 663–677, 2021.
[25] H. G. Bock and K. J. Plitt, A multiple shooting algorithm for direct solution of optimal control problems, *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.
[26] J. Chen, W. Zhan, and M. Tomizuka, Constrained iterative LQR for on-road autonomous driving motion planning, in *Proc. 2017 IEEE 20th International Conference on Intelligent Transportation Systems* (*ITSC*), Yokohama, Japan, 2017, pp. 1–7.
[27] E. F. Camacho and C. B. Alba, *Model Predictive Control*. London, UK: Springer, 2013.
[28] A. Liniger, A. Domahidi, and M. Morari, Optimization-based autonomous racing of 1: 43 scale RC cars, *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.
[29] B. Houska, H. J. Ferreau, and M. Diehl, ACADO toolkit—An open-source framework for automatic control and dynamic optimization, *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.
[30] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, QpOASES: A parametric active-set algorithm for quadratic programming, *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.
[31] S. Thrun, Probabilistic robotics, *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.
[32] I. Draganjac, T. Petrović, D. Miklić, Z. Kovačić, and J. Oršulić, Highly-scalable traffic management of autonomous industrial transportation systems, *Robotics and Computer-Integrated Manufacturing*, vol. 63, p. 101915, 2020.
[33] D. Lam, C. Manzie, and M. C. Good, Model predictive contouring control, in *Proc. 49th IEEE Conference on Decision and Control* (*CDC*), Atlanta, GA, USA, 2010, pp. 6137–6142.



Jianbin Xin received the BSc degree in electrical engineering from Xidian University in 2007 and the MSc degree in control science and engineering from Xi'an Jiaotong University in 2010 in China. In 2015, he received the PhD degree from Delft University of Technology, the Netherlands. Currently, he is an associate professor in the School of Electrical and Information Engineering at Zhengzhou University, China. His research interests include planning and control of smart logistics systems and cooperative robots.

Yaoguang Qu received the bachelor degree in automation from Henan University of Technology, Zhengzhou, China in 2019. Then he received the MSc degree in control engineering from Zhengzhou University, Zhengzhou, China. His research interest focuses on path planning of multiple robots for manufacturing.

Fangfang Zhang received the BSc degree in applied mathematics, the MSc degree in applied mathematics, and the PhD degree in control science and engineering from Shandong University, Jinan, China in 2008, 2011, and 2015, respectively. From 2018 to 2019, he was a visiting scholar with Chinese University of Hong Kong, Hong Kong, China. Currently, he is an associate professor in the School of Electrical and Information Engineering at Zhengzhou University, Zhengzhou, China. His research interests include optimal control of multiagent systems, multirobot formation, and machine vision.

Rudy Negenborn received the MSc degree in computer science from Utrecht University in 2003, and the PhD degree from Delft University of Technology in 2007, both in the Netherlands. He is now a full professor in the Department of Marine and Transport Technology and the head of the Section " Transport Engineering & Logistics" at Delft University of Technology. His fundamental research interests are in the areas of distributed control, multi-agent systems, model predictive control, and optimization.