Complex & Intelligent Systems (2024) 10:397–412
https://doi.org/10.1007/s40747-023-01169-2

ORIGINAL ARTICLE

Online motion planning of mobile cable-driven parallel robots for autonomous navigation in uncertain environments

Jiajun Xu[1] · Byeong-Geon Kim[1] · Xiguang Feng[1] · Kyoung-Su Park[1]

Received: 30 March 2023 / Accepted: 29 June 2023 / Published online: 24 July 2023

Abstract

Mobile cable-driven parallel robots (MCDPRs) offer expanded motion capabilities and workspace compared to traditional cable-driven parallel robots (CDPRs) by incorporating mobile bases. However, additional mobile bases introduce more degree-of-freedom (DoF) and various constraints to make their motion planning a challenging problem. Despite several motion planning methods for MCDPRs being developed in the literature, they are only applicable to known environments, and autonomous navigation in unknown environments with obstacles remains a challenging issue. The ability to navigate autonomously is essential for MCDPRs, as it opens up possibilities for the robot to perform a broad range of tasks in real-world scenarios. To address this limitation, this study proposes an online motion planning method for MCDPRs based on the pipeline of rapidly exploring random tree (RRT). The presented approach explores unknown environments efficiently to produce high-quality collision-free trajectories for MCDPRs. To ensure the optimal execution of the planned trajectories, the study introduces two indicators specifically designed for the mobile bases and the end-effector. These indicators take into account various performance metrics, including trajectory quality and kinematic performance, enabling the determination of the final following trajectory that best aligns with the desired objectives of the robot. Moreover, to effectively handle unknown environments, a vision-based system utilizing an RGB-D camera is developed, allowing for precise MCDPR localization and obstacle detection, ultimately enhancing the autonomy and adaptability of the MCDPR. Finally, the extensive simulations conducted using dynamic simulation software (CoppeliaSim) and the on-board real-world experiments with a self-built MCDPR prototype demonstrate the practical applicability and effectiveness of the proposed method.

Keywords Mobile cable-driven parallel robot · Autonomous navigation · Rapidly exploring random tree · Stability and kinematic performance · Vision system

Introduction

Cable-driven parallel robots (CDPRs) have emerged as a promising class of robotic devices for a wide range of industrial and service applications. Typically, CDPRs are parallel robotic manipulators that employ multiple flexible cables to control the motion of a moving platform (or end-effector). The cables are wound around fixed motorized winch systems to control the position of the end-effector by adjusting their lengths. CDPRs can provide several advantages such as low inertia, large workspace, and high load capacity making them suitable for a variety of tasks of such robots in large-scale construction [1], painting [2] and rehabilitation mechanisms [3].

In spite of the promising performance of CDPRs in numerous applications, several challenges still remain. One of the major drawbacks of classical CDPR is the fixed cable layout, i.e, fixed pulley position and winch system which significantly increases the probability of collision between the cable and the surrounding environment, thereby the available workspace of CDPRs was reduced [4]. Moreover, the fixed cable layout leads to the position of the pulleys must be

✉ Kyoung-Su Park
pks6348@gachon.ac.kr

Jiajun Xu
jilinxujiajun@163.com

Byeong-Geon Kim
bk0725@naver.com

Xiguang Feng
fengx@gachon.ac.kr

1 Department of Mechanical Engineering, Gachon University, 1342 Seongnamdaero, Sujeong-gu, Seongnam-si, Gyeonggi-do 461-701, Korea

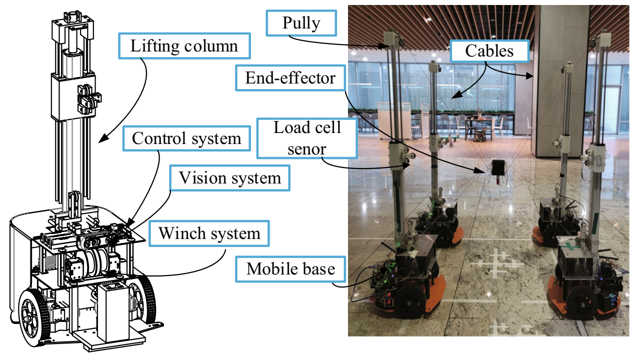


Fig. 1 A self-built MCDPR prototype with a vision system

carefully optimized in order to maximize the workspace [5]. Consequently, Reconfigurable Cable-Driven Parallel Robots (RCDPRs) have been developed by researchers in which cable robots have the ability to change their configurations [6–8]. However, the reconfigurability of most existing RCDPRs is limited due to they still require manually adjusted cable layouts and the base frame cannot be freely moved.

To achieve autonomous reconfigurability of RCDPRs, mobile cable-driven parallel robots (MCDPRs) were developed by researchers [9], in which the additional mobile bases were introduced to mobilize the cable layout and the base frame. The MCDPR developed in this work is shown in Fig. 1, it consists of a classical CDPR with eight cables and a six degree-of freedom (DoF) end-effector mounted on four mobile bases. A mobile base consists of a differential wheeled robot with three wheels and a lifting column. Moreover, an RGB-D camera (Intel RealSense$^{\text{TM}}$ Depth Camera D435i) is mounted on the mobile base to sense the environment. However, additional mobile bases involve complex constraints and high-dimensional configuration space, resulting in motion planning of MCDPRs becoming a challenging issue and requiring to be addressed.

Accordingly, sampling-based motion planning methods have demonstrated promising performance in addressing high-dimensional problems and accommodating various constraints by randomly sampling the configuration space [10]. One of the representative sampling-based path planning algorithms is the Rapidly exploring Random Tree (RRT) method, and some variants of RRT such as RRT* and RRT-connect are proposed [11–13]. In recent years, researchers have developed different RRT-based online motion planning methods for a wide range of robotic systems, including self-driving cars, unmanned aerial vehicles (UAVs), and unmanned ground vehicles (UGVs). Sotirios et al. [14] developed an RRT*-NH method to generate real-time paths for nonholonomic self-driving cars in quasi-static unknown environments. Lin et al. [15] developed a closed-Loop RRT method for UAV dynamic obstacle collision avoidance. Wen et al. [16] developed a heuristic dual sampling domain reduction-based RRT* method to complete the online planning of an unmanned surface vehicle (USV). However, MCDPR consists of multiple mobile bases and CDPR, presenting a unique challenge compared to the aforementioned conventional robotic systems. The mobile bases collaborate and coordinate their movements to achieve a common objective, which involved multi-agent motion planning problem, and high-dimensional state spaces are required to be considered.

One of the main advantages of the RRT algorithm is that it can be easily extended to multidimensional spaces [17]. Zhang et al. [16] proposed an optimization-based map exploration strategy by extending the RRT algorithm, enabling multiple robots to actively explore and construct environment maps. In a similar vein, Lau et al. [18] introduced a temporal memory-based RRT (TM-RRT) exploration strategy designed for multi-robot systems to execute robust exploration tasks within unknown environments. Additionally, Neto et al. [19] presented a Multi-agent Rapidly-exploring Pseudo-random Tree (MRPT) approach, which combines rapid exploration and pseudo-random tree construction, providing real-time motion planning and control for underactuated robots. While the aforementioned methods have showcased their adaptability and potential in multi-robotic systems, the CDPR brings forth additional constraints that necessitate careful consideration. Specifically, in the context of CDPR, there are unique challenges related to cable interference and the intricate kinematics of the robots, which are required to be considered in the present study.

In the pursuit of generating safe trajectories for CDPRs, an adaptive RRT method was proposed in [20] to facilitate the moving obstacle avoidance of moving obstacles. Xiang et al. [21] developed an RRT-based dynamic trajectory planning method specifically tailored for three degrees-of-freedom (DOFs) suspended CDPRs. Mishra et al. [22] introduced the AFG-RRT* method, which considers multiple performance constraints to generate optimized paths for CDPRs. However, these studies only generate feasible trajectories for classic CDPRs. In the case of MCDPRs, Rasheed et al. [23] proposed a sampling-based algorithm that generates collision-free paths while maximizing the wrench capability. Furthermore, the Direct Transcription method was proposed in [24] to provide an optimized path for mobile bases and the end-effector by minimizing the cost function with a series of given constraints. However, the scope of previous MCDPR studies is restricted to scenarios where the environmental information is known a priori. Most recently, Liu et al. [25] introduced a novel approach based on reinforcement learning (RL) to address dynamic obstacle avoidance in real time for CDPRs with mobile bases, further expanding the possibilities for motion planning in the context of MCDPRs.

In this work, we propose an online motion planning method for navigating Mobile Cable-Driven Parallel Robots

(MCDPRs) in uncertain environments. Our approach utilizes RGB-D camera data and introduces a novel multi-time-based RRT algorithm. While recent learning-based approaches have gained attention, they suffer from computational complexity and limited transferability to different environments [26, 27]. Additionally, optimal cascade control for parallel robot platforms has been developed in prior works [28–30], where heuristic algorithms were employed. We also highlight the potential use of a novel exploration-exploitation-based policy for model-free control [31]. However, to simplify the analysis, we have designed two controllers based on the kinematic models of the mobile base and CDPR. The specific contributions of this paper are as follows:

(1) An online motion planning algorithm for MCDPR navigation in real-world scenarios. To address the diverse constraints inherent to MCDPRs and ensure real-time performance, our developed approach employs a continuously constructed time-based tree for the four mobile bases through forward simulation. Consequently, a set of reference trajectories can be generated, allowing for effective motion planning of the MCDPR system.
(2) The novel heuristic functions were designed to determine the final following trajectory for MCDPR in each time interval. These heuristic functions take into account multiple performance metrics, including kinematics stability and distance to the goal, to ensure the generation of a favorable trajectory. Once the trajectories for the four mobile bases have been obtained, we employ a developed partial sampling method to generate the trajectory for the end-effector. This approach allows for the efficient planning of the MCDPR's motion while considering the unique requirements and characteristics of the end-effector.
(3) A vision-based system is developed for MCDPR localization and detecting obstacles by using the RGB-D camera. The system incorporates the proposed method, and we conducted real-world experiments to validate its performance. The experimental results demonstrate the compelling performance of the proposed approach, highlighting its efficacy in accurately localizing the MCDPR and effectively detecting obstacles in a real-world environment.

The remainder of this paper is organized as follows: Robot parameterization and problem formulation are introduced in Sect. "System Modeling". The kinematics and stability model for MCDPRs is established in Sect. "Method Description". In Sect. "Method Validation", the proposed path planning method for MCDPRs is introduced in detail. The method validation is carried out in Sect. "Conclusion and Future Work". Finally, Section 6 concludes the research work.

System modeling

MCDPR parameterization

As shown in Fig. 2, the MCDPR studied in this work consists of $m = 8$ cables and $p = 4$ mobile bases. For j-th mobile base, denoted as $\mathscr{M}_j$, $j = 1, ..., 4$ carries two cables ($m_j = 2$) and has three wheels ($c = 3$) including one caster wheel and two driven wheels. The cable exit points are dependent on the position and orientation of the mobile base. The origin of the global coordinate system is represented as O_0 and the local coordinate system located at the center of the end-effector is denoted as O_e. The mobile bases enable autonomous reconfiguration of the CDPR within a plane parallel to the ground. It should be noted that the mobile base operates as a planar robot, thus limiting the reconfiguration of the plane. As a result, the height of the cable exit points remains constant.

In this study, cables are treated as massless line segments for simplicity. Let i-th cable exit point and anchor point associated with $\mathscr{M}_j$ denote as A_{ij} and B_{ij}, respectively. Moreover, A_{ij} is designed to lie on the axis ${}^{pj}z$ of $\mathscr{M}_j$. Based on the properties of parallel robots, i-th cable vector attached onto $\mathscr{M}_j$, $\boldsymbol{l}_{ij}$, can be expressed as follows

$$\boldsymbol{l}_{ij} = \boldsymbol{a}_{ij} - \boldsymbol{P} - \boldsymbol{R} \cdot {}^{e}\boldsymbol{b}_{ij} \tag{1}$$

where $\boldsymbol{a}_{ij}$ is the position vector of A_{ij} in O_0 coordinate system. ${}^{e}\boldsymbol{b}_{ij}$ represents the position vector of B_{ij} in the moving coordinate system O_e. $\boldsymbol{P}$ and $\boldsymbol{R}$ is a three-dimensional position vector and rotation matrix of the end-effector, respectively. Hence, the velocity relationship between the cable vector and the end-effector can be obtained via the differentiation of (1) with respect to time

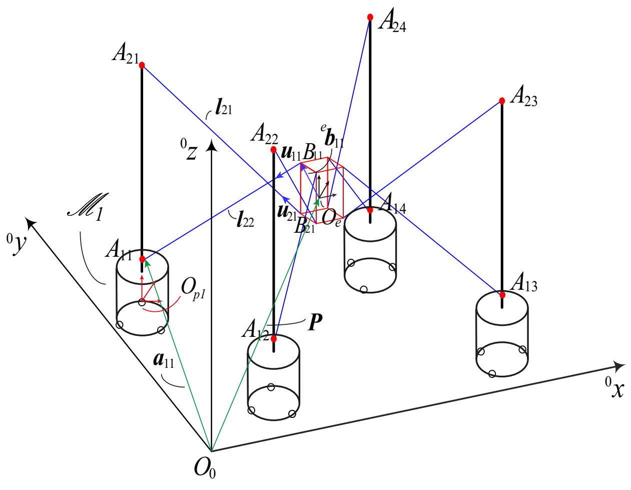


Fig. 2 Structural diagram of MCDPR

$$\dot{\boldsymbol{l}}_{i,j} = \text{-}\boldsymbol{J}^T \dot{\boldsymbol{P}} \tag{2}$$

$$\boldsymbol{J}(\boldsymbol{X}) = \begin{bmatrix} \boldsymbol{u}_{1,1} & \cdots & \boldsymbol{u}_{2,j} \\ \boldsymbol{u}_{1,1} \times (\boldsymbol{R} \cdot {}^e\boldsymbol{b}_{1,1}) & \cdots & \boldsymbol{u}_{2,j} \times (\boldsymbol{R} \cdot {}^e\boldsymbol{b}_{m,j}) \end{bmatrix} \tag{3}$$

where $\boldsymbol{J} \in \mathbb{R}^{n \times m}$ is the Jacobian matrix of the carried CDPR. $\boldsymbol{u}_{i,j}$ is the unit vector of $\boldsymbol{l}_{ij}$. In this work, we limit the end-effector to perform only translational motion. By using (1) and (3), the position and the velocity of the end-effector can be controlled by varying the length of the cable.

Problem formulation

To accomplish autonomous navigation of MCDPR in unknown environments, the sampling-based motion planning method requires searching the state space to construct a graph. We first define the state of j-th mobile base and the end-effector at k-th time step is $\boldsymbol{m}_{j,k} = [{}^0x_{j,k}, {}^0y_{j,k}]$ and $\boldsymbol{P}_k = [{}^0x_k, {}^0y_k, {}^0z_k]$, $k = 1, ..., N$, respectively. The system state was constructed by multiple mobile bases at kth time step is $\boldsymbol{M}_k = [\boldsymbol{m}_{1,k}^T, \boldsymbol{m}_{2,k}^T, \boldsymbol{m}_{3,k}^T, \boldsymbol{m}_{4,k}^T]^T$. Therefore, the system state of MCDPR at kth time step can be expressed as $\boldsymbol{X}_k = [\boldsymbol{P}_k^T, \boldsymbol{M}_k^T]^T$.

The MCDPR involves planning the movement of multiple mobile bases in a constrained environment with internal and external constraints. To simplify the computation, the mobile bases and obstacles are modeled as cylindrical structures. At each time step, the path planning algorithm takes into account the following constraints for the jth mobile base:

$$\left\|\boldsymbol{m}_{h,k} - \boldsymbol{m}_{j,k}\right\| > L_m, \text{ for } h = 1, .., 4, \ h \neq j \tag{4}$$

$$\left\|\boldsymbol{m}_{j,k} - \boldsymbol{o}_q\right\| > r_m + r_o, \text{ for } q = 1, .., s, \tag{5}$$

$$\beta_{j,k} = \cos^{-1} \frac{\boldsymbol{d}_{j,k-1} \cdot \boldsymbol{d}_{j,k}}{\|\boldsymbol{d}_{j,k-1}\| \|\boldsymbol{d}_{j,k}\|} < \beta_{\max} \tag{6}$$

where (4)–(6) impose the distance and angle constraints of mobile bases. Equation (4) imposes a distance constraint between the current mobile base position $\boldsymbol{m}_{j,k}$ and the positions of other mobile bases $\boldsymbol{m}_{h,k}$, ensuring a minimum separation distance L_m between them and promotes safe navigation for the MCDPR. Equation (5) represents a distance constraint between the mobile base position $\boldsymbol{m}_{j,k}$ and the positions of obstacles $\boldsymbol{o}_q$, which ensures that the distance between the mobile base and any obstacle is greater than the sum of their respective radii, r_m and r_o, to prevent collision between the MCDPR and obstacles. To address the nonholonomic constraints, the algorithm uses directional vectors $\boldsymbol{d}_k = \boldsymbol{m}_{j,k} - \boldsymbol{m}_{j,k-1}$ in Eq. (6), which represent the movement of the mobile base between adjacent states $\mathscr{M}_j$. The algorithm ensures smooth and continuous movement of mobile bases by imposing a limit on the maximum turning angle, denoted by $\beta_{\max}$. Moreover, due to the limited workspace and cable length of the CDPR, the end-effector requires to satisfy the following constraints

$$\|\boldsymbol{c}_k - \boldsymbol{b}_k\| > L_c \tag{7}$$

$$l_{\min} \leq \left\|[\boldsymbol{m}_{j,k}{}^T \ h_i]^T - \boldsymbol{P}_k\right\| \leq l_{\max}, \text{ for } i = 1, 2 \tag{8}$$

$$\boldsymbol{P}_k \in \mathcal{W}_k \tag{9}$$

where $\boldsymbol{c_k}$ and $\boldsymbol{b_k}$ denotes the two closest point between cables and obstacles at kth time step, respectively. Therefore, Eq. (7) ensures a safe distance L_c between the cables and obstacles, preventing collisions and maintaining the safety of the MCDPR system. Equation (12) imposes a constraint on the cable lengths by limiting them to lie within the range $l_{\min}$ and $l_{\max}$, which ensures that the cable lengths are within acceptable limits and facilitates the proper functioning of the MCDPR. Moreover, Eq. (9) specifies that the end-effector's position $\boldsymbol{P}_k$ should lie within the defined workspace $\mathcal{W}_k$ to ensure the end-effector to operate within the designated workspace, allowing the MCDPR to perform its intended tasks effectively.

Let $\mathcal{X}(\boldsymbol{M}, \boldsymbol{P}) \subset \mathbb{R}^{2p+3}$ denotes the state space of MCDPR and the obstacle space $\mathcal{X}_{obs} \subseteq \mathcal{X}$ refers to the states that the robot collides with the obstacle. The free space that satisfy (4)–(9) contraints given by $\mathcal{X}_{free} \subseteq \mathcal{X}/\mathcal{X}_{obs}$. Let Σ_k denote the set of feasible trajectories obtained by the proposed method at time k. The autonomous navigation problem for MCDPR can be defined by

$$\begin{aligned} &\sigma^* = \underset{\sigma \in \Sigma_k}{\operatorname{argmin}}\, c(\sigma) \\ &\text{s.t. } \sigma(0) = \mathcal{X}_{init} \\ &\quad \sigma(t_f) \in \mathcal{X}_{goal} \\ &\quad \sigma(t) \in \mathcal{X}_{free}(t), \forall t \ \in \ [0, t_f]. \end{aligned} \tag{10}$$

where $\mathcal{X}_{\text{init}}$ and $\mathcal{X}_{\text{goal}}$ denote the initial and goal state of mobile bases. $c : \Sigma \mapsto \mathbb{R} \geq 0$ is the defined cost function for MCDPR. In the next section, we will discuss the details of the aforementioned problem and propose the sampling-based motion planning method to solve it in real-time.

Method description

We now present the sampling-based method to deal with the autonomous navigation problem for MCDPR. The proposed method utilizes RRT-based approach to generate feasible trajectories at each time step by constructing a tree-structured graph, denoted by $\mathcal{T}(\mathcal{V}, \mathcal{E})$, where V is the set of nodes representing the state of mobile bases $\boldsymbol{M}$ and $\mathcal{E}$ is the set of edges connecting them with state transition. The proposed method is summarized in Algorithm 1.

In Algorithm 1, the complete loop of the suggested method includes several steps or stages that are carried out in a spe-

Algorithm 1 Multi Time-based RRT (MTB-RRT)

```
Input: $\mathcal{X}_{init}, \mathcal{X}_{goal}, \mathcal{X}_{obs}$
Output: $\sigma^*$
 1: Check the feasibility of the initial state
 2: $\mathcal{X}_0 \leftarrow \mathcal{X}_{init}$
 3: Initialize tree $\mathcal{T} \leftarrow \mathcal{X}_0$
 4: while goal not reached do
 5:     Observe the current state of MCDPR $\mathcal{X}_k(\boldsymbol{M}_k, \boldsymbol{P})$
 6:     Observe the state of obstacle $\mathcal{X}_{obs,k}$ by using vision sensor.
 7:     $t_0 \leftarrow clock()$
 8:     while $clock() \leq t_0 + \Delta t$ do
 9:         $\mathcal{T}_k \leftarrow$ MultiTreeExtend($\mathcal{T}_{k-1}$,$\mathcal{X}_{obs,k}$,$\mathcal{X}_{goal}$, $Q$)
10:     end while
11:     $\boldsymbol{M}_k^P \leftarrow$ BackPropagation($\mathcal{T}_k$)
12:     $\boldsymbol{M}_k^{P^*} \leftarrow$ FindCheapest($\boldsymbol{M}_k^P$)
13:     for each $\boldsymbol{M}_k \in \boldsymbol{M}_k^{P^*}$ do
14:         $\boldsymbol{P}_k^M \leftarrow$ GridSampling($\boldsymbol{M}_k$)
15:         $\boldsymbol{P}_k^* \leftarrow$ FindBest($\boldsymbol{P}_k^M$)
16:         $\boldsymbol{P}_k^{P^*} \leftarrow \boldsymbol{P}_k^{P^*} \cup \boldsymbol{P}_k^*$
17:     end for
18:     $\sigma_k^* \leftarrow \boldsymbol{M}_k^{P^*} \cup \boldsymbol{P}_k^{P^*}$
19:     if $\sigma_k^*$ is empty then
20:         Brake()
21:     else
22:         $\mathcal{T}_{k+1} \leftarrow$ UpdateTree($\mathcal{T}_k$,$\sigma_k^*$)
23:     end if
24:     $t_0 \leftarrow clock()$
25:     Send control commands to MCDPR to follow the trajectory $\sigma_k^*$
26: end while
```

cific sequence. Checking the feasibility of the initial state is the first step in this process, and it is essential for ensuring that the method can proceed successfully. If the initial state satisfies (4)–(9), the initial state has been assigned as the root of the tree $\mathcal{T}$. If the goal point has not been reached, we observe the current state of MCDPR $\mathcal{X}_k(\boldsymbol{M}_k, \boldsymbol{P})$, which can be done by using vision-based simultaneous localization and mapping (SLAM) technique and also the state of obstacles can be estimated using RGB-D camera. Based on the observed state of both MCDPR and the environment, the tree $\mathcal{T}_k$ is constructed during limited time step Δt. After the tree has been constructed, a series of feasible trajectories of mobile bases $\boldsymbol{M}_k^P$ are obtained by continuously propagating the sub-node of the parent node and the optimized trajectories with minimum cost, $\boldsymbol{M}_k^{P^*}$, can be generated.

Consequently, each state in $\boldsymbol{M}_k^{P^*}$ represents a specific configuration of four mobile bases and is denoted as $\boldsymbol{M}_k$. For each $\boldsymbol{M}_k$, a sequence of potential end-effector positions $\boldsymbol{P}_k^M$ are sampled and the best end-effector position $\boldsymbol{P}_k^*$ is obtained by minimizing the given cost. By continuously accumulating nodes $\boldsymbol{P}_k^*$, the final path of the end-effector $\boldsymbol{P}_k^{P^*}$ can be generated, and the trajectory for MCDPR that minimizes a given cost function σ_k^* can be found. In cases where the set σ_k^* is found to be empty, indicating the absence of a feasible trajectory, the Brake() function is invoked to bring the robot to a halt until a new viable trajectory can be generated. This mechanism ensures that the robot does not proceed with an invalid or unsafe trajectory, thereby preventing any undesired actions or potential hazards. Otherwise, the tree will be updated to obtain a new tree $\mathcal{T}_{k+1}$ for the next loop. Finally, the MCDPR is controlled to follow σ_k^* for one-time step.

The proposed tree construction method was illustrated in Fig. 3. For more clarity and to avoid overlap, we only show two trees associated with $\mathcal{M}_1$ and $\mathcal{M}_2$, in which ${}^q\boldsymbol{M}_{i,j}$ represent the qth node on the ith tree of jth mobile base. For every time interval, the feasible tree is constructed shown in Fig. 3a, and the depth of the tree is limited to accommodate the unknown environment. Hence, a series of feasible trajectories can be generated by continuously propagating from the leaf node to the root node. As shown in Fig. 3b, five feasible trajectories can be found and the final following trajectory is obtained by using the cost function. During the MCDPR follow the trajectory in one step, the tree is pruned and unnecessary branches are removed for growing a new tree, as shown in Fig. 3c.

Multiple tree extension and rewiring

In this subsection, the tree extension algorithm is presented in detail. As shown in Algorithm 2, the feasibility of the remained tree $\mathcal{T}_k$ is checked. In certain situations where the environment significantly changes, some branches of the algorithm may not be feasible or practical to execute, which can negatively impact the real-time performance of the algorithm.

After the feasibility of the $\mathcal{T}_k$ is checked, the algorithm proceeds in a loop, where in each iteration, it generates a new state $\boldsymbol{M}_{\text{new}}$ for the robot using adaptive sampling, which is a sampling strategy that attempts to bias the sampling towards unexplored areas of the state space. As a result, for the case of four mobile bases, we only need to sample an 8-dimensional vector, and for each subtree comprising the mobile bases, a two-dimensional vector suffices for sampling. This dimension reduction enables the acquisition of valid nodes within small time intervals, addressing the challenges associated with handling such high-dimensional data in the algorithm. Then, the algorithm attempts to connect this new state to the nearest node in the tree $\boldsymbol{M}_{\text{nearest}}$ using a steering function, which generates a path from the nearest node to the new state. Moreover, $\boldsymbol{M}_{\text{new}}$ are obtained with a fixed expansion angle $\boldsymbol{M}_{\text{new}} = \boldsymbol{M}_{\text{near}} + \eta * [\cos(\theta), \sin(\theta)]$. If $\boldsymbol{M}_{\text{new}}$ is feasible by checking constraints (Eqs. 4–9), then the new state is added to the tree as a new node, and an edge is added to connect it to the nearest node.

Furthermore, the heuristic method of Algorithm 2 was introduced, which aims to optimize the path by utilizing neighbor vertices. After generating a feasible new node $\boldsymbol{M}_{\text{new}}$, its neighbor vertices were obtained, denoted as $\mathcal{M}_{\text{near}}$, by employing a search radius r^*. Specifically, each $\boldsymbol{m}_{\text{near},j}$ that is directly connected to $\boldsymbol{m}_{\text{new},j}$ and attempts to minimize

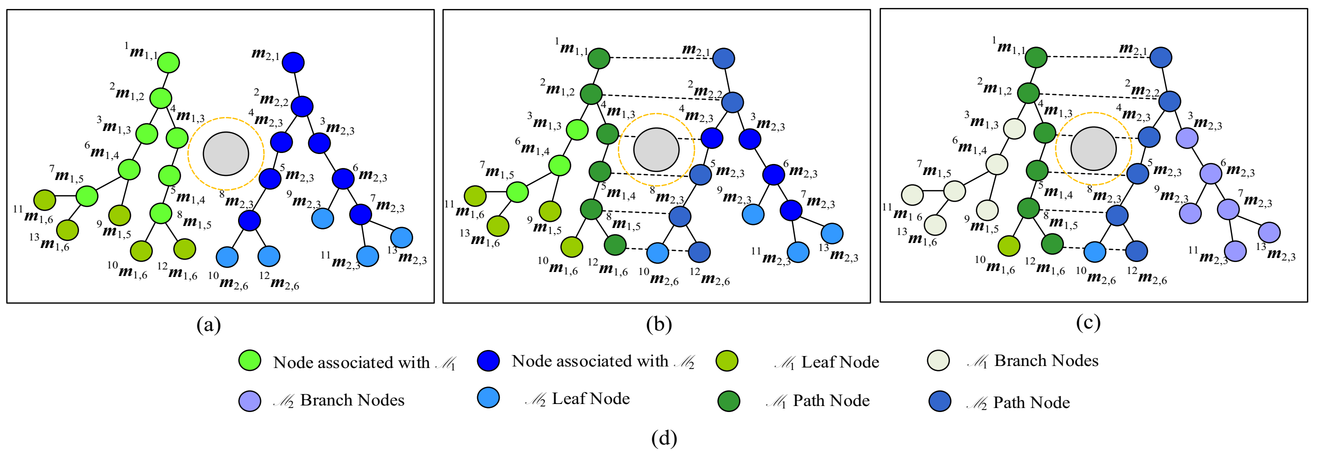

(a)
(b)
(c)
(d)
Node associated with $\mathcal{M}_1$
Node associated with $\mathcal{M}_2$
$\mathcal{M}_1$ Leaf Node
$\mathcal{M}_1$ Branch Nodes
$\mathcal{M}_2$ Branch Nodes
$\mathcal{M}_2$ Leaf Node
$\mathcal{M}_1$ Path Node
$\mathcal{M}_2$ Path Node

Fig. 3 Illustrations of proposed sampling-based path planning method for CDPR moving obstacle avoidance. Every node on the tree is feasible in its time stamp according to the prediction of the moving obstacle. **a** Time-based tree was constructed with a certain depth. **b** Optimal trajectory was obtained from the tree. **c** Delete additional branches and move one step

Algorithm 2 MultiTreeExtend

```
Input: $\mathcal{T}_{k-1}$, $\mathcal{X}_{obs,k}$, $\mathcal{X}_{goal}$, $Q$
Output: $\mathcal{T}_k$
 1: Check the feasibility of $\mathcal{T}_k$
 2: $\boldsymbol{M}_{rand} \leftarrow$ AdaptiveSampling($\boldsymbol{M}_{goal}$,$\boldsymbol{M}_{parent}$,$\mathcal{T}$)
 3: $\boldsymbol{M}_{nearest} \leftarrow$ Nearest( $\boldsymbol{M}_{rand}$,$\mathcal{T}$,$V_j$)
 4: $u_{new}$, $\boldsymbol{M}_{new}$ $\leftarrow$Steer($\boldsymbol{M}_{nearest}$,$\boldsymbol{M}_{rand}$,$\eta$,$\theta$)
 5: if $\boldsymbol{M}_{new}$ is feasible then
 6:     if Depth($\boldsymbol{M}_{new}$) is $Q$ then
 7:         Assign $\boldsymbol{M}_{new}$ as leaf node
 8:         Continue
 9:     else
10:         $\mathcal{M}_{near} \leftarrow$ NearestNeighbors($\boldsymbol{M}_{new}$,$\mathcal{T}$,$r*$)
11:         $c_{min} = h(\boldsymbol{M}_{new}) + d(\boldsymbol{M}_{new}, \boldsymbol{M}_{nearest})$
12:         for each $M_{near} \in \mathcal{M}_{near}$ do
13:             $(V_{new,j}$,$E_{new,j}) \leftarrow$ Connect($\boldsymbol{m}_{near,j}$,$\boldsymbol{m}_{new,j}$)
14:             if Path($\boldsymbol{m}_{near,j}$,$\boldsymbol{m}_{new,j}$) is feasible then
15:                 $c_{new,j} = h(\boldsymbol{m}_{near,j}) + dist(\boldsymbol{m}_{new,j}, \boldsymbol{m}_{near,j})$
16:                 if $c_{new,j} < c_{min,j}$ then
17:                     $c_{min,j} \leftarrow c_{new,j}$,
                        $\boldsymbol{M}_{near}$ $\leftarrow$ReSetParent($m_{near,j}$,$\mathcal{T}$)
18:                     if $\boldsymbol{M}_{near}$ is feasible then
19:                         Continue
20:                     end if
21:                 end if
22:             end if
23:         end for
24:     end if
25:     $\mathcal{T}.\mathcal{V}$ $\leftarrow$InsertVertexToTree($\boldsymbol{M}_{new}$, $\mathcal{T}$)
26:     $\mathcal{T}.\mathcal{E}$ $\leftarrow$InsertEdgeToTree($\boldsymbol{M}_{new}$,$\boldsymbol{M}_{parent}$,$\mathcal{T}$)
27: else
28:     Update $\boldsymbol{M}_{new}$.Failure_rate
29: end if
```

the heuristic cost. Here, $h(\boldsymbol{M}_{\text{new}})$ represents the total path length from the initial node $\boldsymbol{M}_{\text{init}}$ to $\boldsymbol{M}_{\text{new}}$. However, assigning $\boldsymbol{m}_{\text{near},j}$ as the parent node may disrupt the tree structure, leading to vertex disjunction. To address this issue, ReSetParent() method was proposed, which resets the passed vertices on the edge $(\boldsymbol{m}_{\text{near},j}, \boldsymbol{m}_{\text{new},j})$ while preserving the parent node. By adopting this approach, we ensure the stability of the tree structure and maintain the integrity of the graph. The Heuristic method and the ReSetParent() technique are specifically designed to enhance the algorithm's efficiency and reliability. Through careful selection of neighbor vertices and preservation of the tree structure, our algorithm is capable of generating high-quality paths that satisfy the given constraints and objectives.

Finally, the algorithm iteratively executes this process until the time limit is reached. The resulting final tree represents a feasible path from the current root state to the goal state, providing guidance for the robot to traverse along the determined trajectory. This approach ensures that the robot can navigate the environment effectively and reach the desired goal state.

Feasible paths generation for the end-effector

After the RRT tree $\mathcal{T}_k$ is obtained from Algorithm 2, the path for mobile bases $\boldsymbol{M}_k^{P^*}$ can be generated by consecutively finding $\boldsymbol{M}_{\text{parent}}$ from the root of the tree. In this work, the grid sampling method is developed to generate the feasible path for the end-effector, and the proposed method is shown in Algorithm 3.

Algorithm 3 Grid Sampling Method

```
Input: $P^k_P, Q$
Output: $P^M_k$
 1: Check the end-effector current state $M^*_k$
 2: Determine the search area $\mathcal{S}_p$ with radius $r_s$
 3: Discrete $\mathcal{S}_p$ to obtain $Q_p$ cells $\{c_1, ..., c_Q\}$
 4: for $n = 1$ to $Q$ do
 5:     if not feasible $(c_n)$ then
 6:         Continue
 7:     else
 8:         $P^M_k \leftarrow P^M_k \cup c_n$
 9:     end if
10: end for
```

The Algorithm 3 takes two inputs: $\boldsymbol{P}^k_P$ and Q. $\boldsymbol{P}^k_P$ represents the current state of an end-effector, which denotes the three-dimensional coordinates of the end-effector. Q is a user-defined parameter that specifies the number of cells to be generated in the search area. The algorithm first checks the current state of the end-effector, denoted as $\boldsymbol{M}k^*$. This step is necessary to determine the starting position for the search. The algorithm then determines the search area, denoted as $\mathcal{S}p$, which is a circular region around the current end-effector position with a radius of r_s. Next, the algorithm discretizes the search area into Q cells, denoted as $c_1, ..., c_Q$. Each cell represents a possible location for the end-effector. The algorithm then iterates over each cell to determine if it is a feasible location for the end-effector to move to. If a cell is not feasible, the algorithm skips it and moves on to the next cell. If a cell is feasible, it is added to the output set $\boldsymbol{P}^M_k$, which contains all the feasible locations for the end-effector to move to. Finally, the algorithm returns $\boldsymbol{P}^M_k$ as the output.

Cost function

In this subsection, the cost functions for generating the optimal path for the mobile bases and the end-effector are presented. For mobile bases, the FindCheapest$(\boldsymbol{M}^P_k, \mathcal{X}_{\text{goal}}, \mathcal{X}_{\text{obs}})$ function in Algorithm 1 was used to obtain the optimal paths for mobile bases by minimizing the following cost function c_m, which consists of three terms, considering the state error,

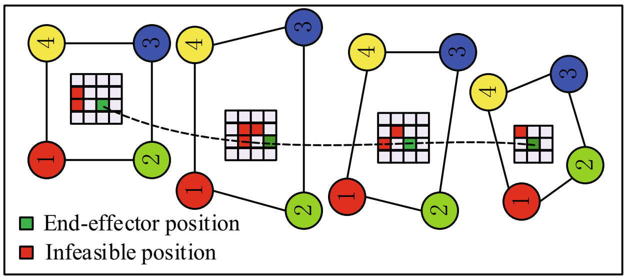


Fig. 4 illustration of the path planning for end-effector

potential collision avoidance, and the distance to the target.

$$c_m = \sum_{k=1}^{Q} J_r(\boldsymbol{M}^p_k, \boldsymbol{M}^p_{k_{\text{ref}}}) + J_c(\boldsymbol{M}^p_k, \mathcal{X}_{\text{obs}}) + J_g(\boldsymbol{M}^p_l, \mathcal{X}_{\text{goal}}) \tag{11}$$

Where the first term $J_r(\boldsymbol{M}^p_k, \boldsymbol{M}^p_{k_{\text{ref}}})$ penalizes the deviation of the predicted state $\boldsymbol{M}^p_k$ from the desired state vector $\boldsymbol{M}^p_{k_{\text{ref}}}$ in a quadratic sense. It measures the difference between these two states using the Q_m-weighted Euclidean norm, encouraging the mobile bases to closely follow the desired trajectory as shown follows

$$J_r(\boldsymbol{M}^p_k, \boldsymbol{M}^p_{k_{ref}}) = \|\boldsymbol{M}^p_k - \boldsymbol{M}^p_{k_{\text{ref}}}\|^2_{Q_m} \tag{12}$$

The collision cost term in (11), $J_c(\boldsymbol{M}^p_k, \mathcal{X}_{\text{obs}})$, is shown in (13) and designed to avoid collisions with obstacles. It is computed as a sum over all obstacles, where each term considers the distance between the predicted state $\boldsymbol{M}^p_k$ and a specific obstacle. This term utilizes the logistic function to create a smooth and bounded cost function, with $Q_{c,q}$ and k_q as tuning parameters controlling its shape and sensitivity.

$$J_c = \sum_{q=1}^{s} \frac{Q_{c,q}}{1 + \exp k_q(\left\|\boldsymbol{M}^p_k - \boldsymbol{o}_q\right\| - r_m - r_o)} \tag{13}$$

Moreover, the cost-to-go function is given by $J_g(\boldsymbol{M}^p_l, \mathcal{X}_{\text{goal}})$ can be expressed in (14). aims to guide the mobile bases to reach the goal state quickly. It measures the distance between the leaf node $\boldsymbol{M}^p_l$ in the path $\boldsymbol{M}^p_k$ and the goal state $\mathcal{X}_{\text{goal}}$. The G_m-weighted Euclidean norm is used to compute this cost, providing a measure of how close the mobile bases are to reaching the goal.

$$J_g = \|\boldsymbol{M}^p_l - \boldsymbol{M}_{\text{goal}}\|^2_{G_m} \tag{14}$$

After obtaining the final following path of the mobile bases, denoted as $\boldsymbol{M}^{P*}_k$, the path for the end-effector can be obtained using the GridSampling $(\boldsymbol{M}_k)$ function. The optimal path for the end-effector, denoted as $\boldsymbol{P}^*_k$, is then determined by the FindBest$(\boldsymbol{M}_k)$ function by using following equation

$$J_e = \alpha\gamma_k + \beta\gamma_s \tag{15}$$

where kinematic cost, $\gamma_k = \sqrt{\det(\boldsymbol{J}\boldsymbol{J}^T)}$, measures the manipulability of the robot along the trajectory, which is computed based on the determinant of the Jacobian matrix, providing information about the robot's ability to perform desired tasks effectively. Moreover, The smoothness cost, γ_s, characterizes the smoothness of the trajectory. It evaluates the squared differences between consecutive points in the path,

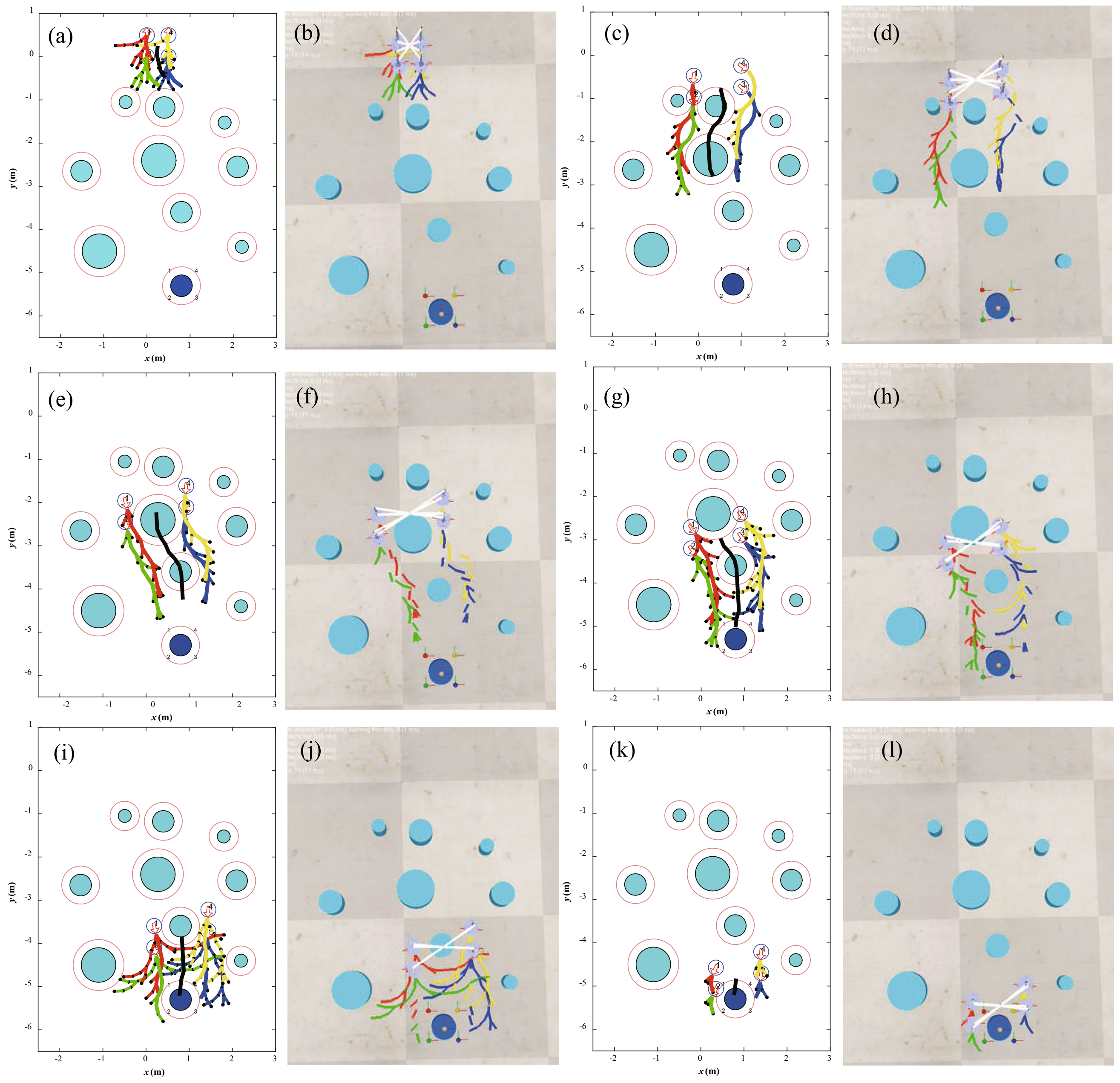


Fig. 5 Results of simulation and verification in CoppeliaSim. The red, green, blue, yellow, and black represent the $\mathscr{M}_1$, $\mathscr{M}_2$, $\mathscr{M}_3$, $\mathscr{M}_4$ and the end-effector **a**, **b** $t = 0$ s. **c**, **d** $t = 6$ s. **e**, **f** $t = 12$ s. **g**, **h** $t = 18$ s. **i**, **j** $t = 24$ s. **k**, **l** $t = 30$ s

$\boldsymbol{P}_{k+1}^M - 2\boldsymbol{P}_k^M + \boldsymbol{P}_{k-1}^M$, where $\boldsymbol{P}k^M$ represents the position of the end-effector at time step k. This term promotes smooth and continuous motion of the end-effector. The weighting factors α and β determine the relative importance of dexterity, obstacle avoidance, and smoothness costs in the overall evaluation. These factors can be adjusted according to the specific requirements of the application, allowing customization of the optimization process based on desired priorities.

Method validation

Initialization set-up and results

Within this section, we present a series of simulations aimed at demonstrating the effectiveness and efficiency of our proposed algorithm. The cluttered environment in which the simulations are performed consists of the dynamic model

Table 1 Simulation parameter

Parameters	Value	Parameters	Value
L_m	0.38 m	$l_{\min}$	0.5 m
r_m	0.2 m	$l_{\max}$	4 m
$\beta_{\max}$	65°	r_c	0.5 m
L_c	0.05 m	$r_{gc,j}$	0.3 m
$r*$	0.5 m	η	0.2 m
α	0.5	β	0.5
Q	10	ω_2	0.5

of our MCDPR prototype, developed through the utilization of the CoppeliaSim (formerly V-REP) robot simulator software [32]. Specifically, we have implemented ten obstacles within the environment, all of which are modeled as cylinders with varying radii of 0.15 m, 0.25 m, and 0.4 m. These obstacles possess an equivalent height of 0.4 m and are randomly positioned throughout the environment. Additionally, five distinct goal points have been designated around the dark blue obstacle, which serves as the target obstacle where the robot executes tasks, such as picking and/or releasing operations.

The proposed method is initialized using the parameters listed in Table 1. To achieve high accuracy during the consecutive state transition from $\boldsymbol{X}_k$ to $\boldsymbol{X}_{k+1}$, a relatively small Δt of 0.5 s is used in this work. If Δt is too large, more nodes can be explored, but there is also a higher likelihood of overskipping or overlooking obstacles in the environment. Moreover, Δt is a constant in this work to maintain the integrity and coherence of the tree structure, as changes in sample spacing may disrupt connectivity and introduce computational complexity. For each $\boldsymbol{M}_k^p$, a set of pulleys' coordinates can be determined using $h_1 = 0.285$ m and $h_2 = 0.926$ m. As a result, the mass center of the mobile base can be directly obtained from the MCDPR dynamic model in CoppeliaSim, which is given by $[0.12, 0.34, 0.36]^T$ with respect to $O_p j$ coordinate system. Therefore, γ_k and γ_s can be computed based on the configuration of the mobile bases.

The results of the simulation depicted in Fig. 5 demonstrate the generation of feasible paths for both the mobile bases and end-effector in each time interval. The final paths for both the end-effector and mobile bases are highlighted in bold and determined by minimizing Eqs. 11 and (15). As the maximum tree length is constrained, it is observed that the tree's predicted distance during growth is limited by assuming the tree is confined by the unknown aspects of the environment and sensor accuracy. Furthermore, in this simulation, we can acquire specific states of the robot and obstacles through the software, which allows for the direct acquisition of feasibility detection within the algorithm. Figure 7 illustrates the variation of the end-effector's z-coordinate. One should note that the end-effector's z-coordinate is determined by using the cost function (15). In order to illustrate the impact of path optimization, we conducted a comparative analysis of the performance change in the end-effector movement with and without optimization. As all obstacles were set to the same height of 0.4 ms, the non-optimized process involved fixing the Z-coordinate of the end-effector at 0.43 m to ensure collision avoidance and the results were shown in Fig. 9. The behavior of J_e with respect to a fixed Z-coordinate exhibits an initial marginal increase followed by a subsequent decline with oscillatory patterns. Conversely, with the implementation of optimization processes, J_e experiences a rapid escalation and oscillates at higher levels. This phenomenon exemplifies the efficacy of the optimization techniques utilized.

To streamline the problem, the end-effector is represented as a single point to facilitate collision detection. Figure 8 depicts the three minimum distances that pertain to mobile bases/mobile bases, mobile bases/obstacles, and cables/obstacles. It is noteworthy that the distance constraints specified by Eqs. (4), (5), and (7) can be met by adhering to the aforementioned distances.

MCDPR verification in CoppeliaSim

In this section, we demonstrate the resulting MCDPR trajectories σ^* in each iteration are simulated in the CoppeliaSim environment to verify the proposed method. The software simulation framework is depicted in Fig. 11, which describes a process for simulating the proposed method for MCDPR. The process begins with the input of parameters required for the simulation. These parameters could include the robot's physical characteristics, the properties of the environment, and the initial conditions of the simulation. Once the parameters have been inputted, the next step is to create the simulation environment and load the MCDPR model in which the simulation will take place. Then, the simulation will continue to run until the robot has reached the target.

In the course of the simulation, the state of the robot is acquired periodically and a path is generated by using the proposed MTB-RRT method for the robot to follow. Next, control commands are computed based on the generated path. Then, control directives are derived based on the path that was generated. To control the mobile bases, the continuous velocity profiles of the mobile bases are transformed into the rotational velocities of the mobile base's wheels via its kinematic model. As demonstrated in Fig. 10, the continuity of the velocity profiles of the mobile bases and the end-effector is guaranteed in each iteration. In addition, to control the end-effector, we use (1) and (2) to regulate the length and speed of the cables shown in Fig. 12. The obtained rotational velocities of the wheels are dispatched to the revolute joints linked with the wheels to control the mobile bases in the simulation

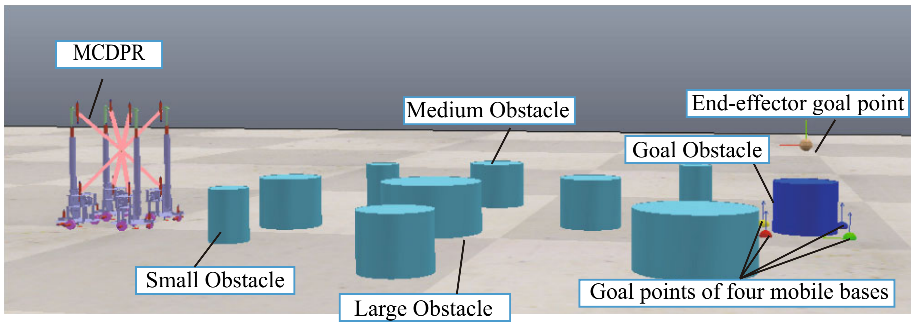


Fig. 6 Simulation environment developed using CoppeliaSim software [32]

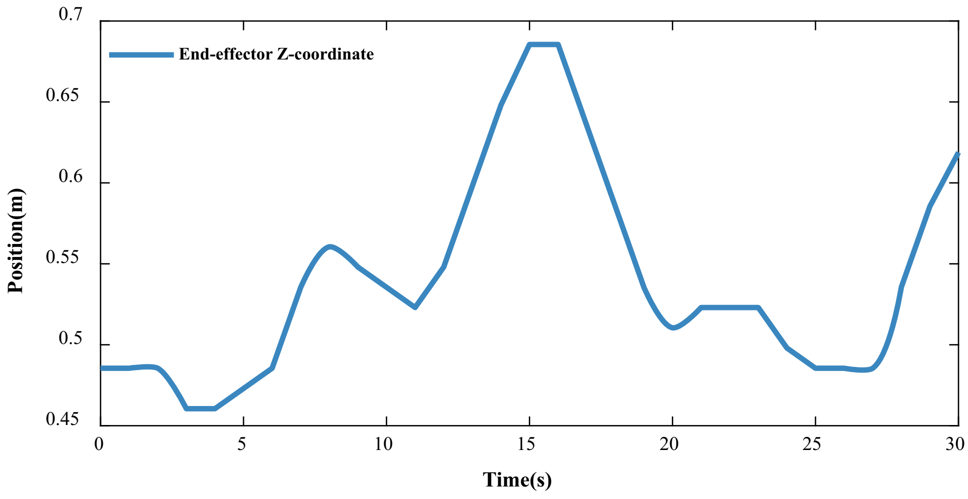


Fig. 7 z-coordinate of the end-effector

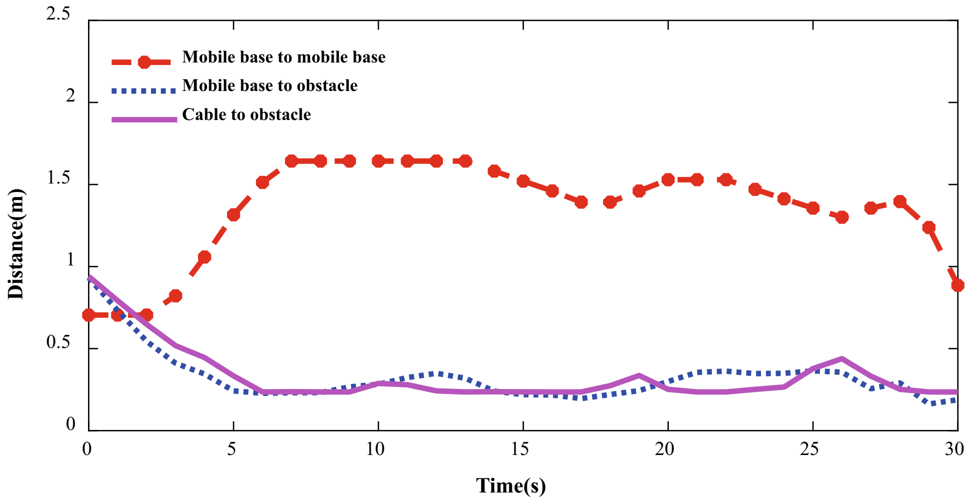


Fig. 8 Relative distance change with time

software CoppeliaSim. Besides, the length of the cable can be regulated via prismatic joints connected with the cables. As a result, the MCDPR model is actuated by these joints. The state of the MCDPR can be directly obtained from the CoppeliaSim throughout the movement, which will be compared with the intended path to obtain the error.

The simulation clips presented in Fig. 5 demonstrate that the MCDPR effectively adheres to desired paths while successfully avoiding collisions. Additionally, Fig. 13 depicts the discrepancy between the simulated MCDPR's actual path and the desired path, revealing a maximum error of roughly 3.8 cm at both the mobile base and end-effector. These findings confirm the validity and stability of the proposed motion planning method. These results demonstrate the feasibility of utilizing the MCDPR for complex tasks requiring precise control and maneuverability in dynamic environments.

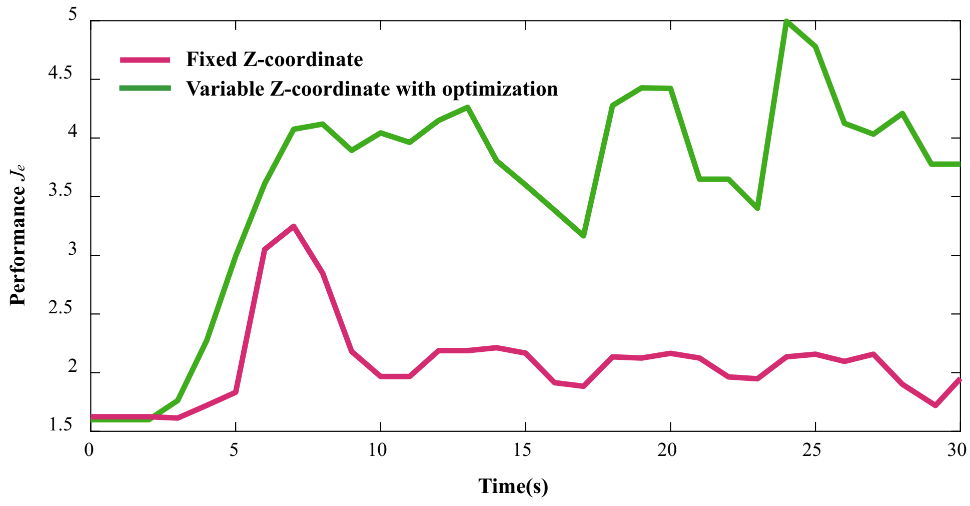


Fig. 9 Performace J_e Comparison

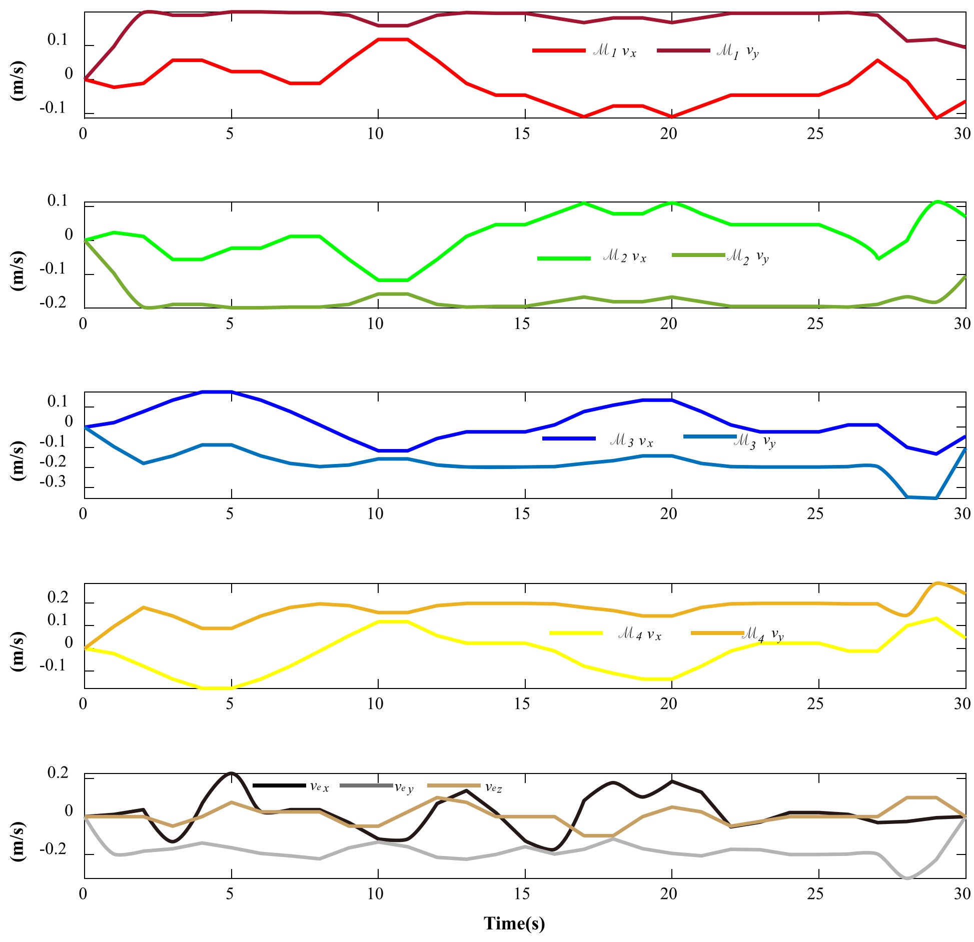


Fig. 10 Velocity profiles of MCDPR during the simulation

Experimential verification

Experimental setup

The proposed approach has been experimentally validated using a self-built MCDPR. A specific scenario was designed, wherein the MCDPR passed through the first obstacle illustrated in Fig. 5. The MCDPR system in our study utilized the NVIDIA Jetson Nano Developer Kit as the control computer, which plays a crucial role in orchestrating the operation of the MCDPR by transmitting the prescribed cable lengths and mobile base velocities necessary for its functioning. To enhance the system's perception capabilities, an RGB-D camera was strategically mounted on the mobile base.

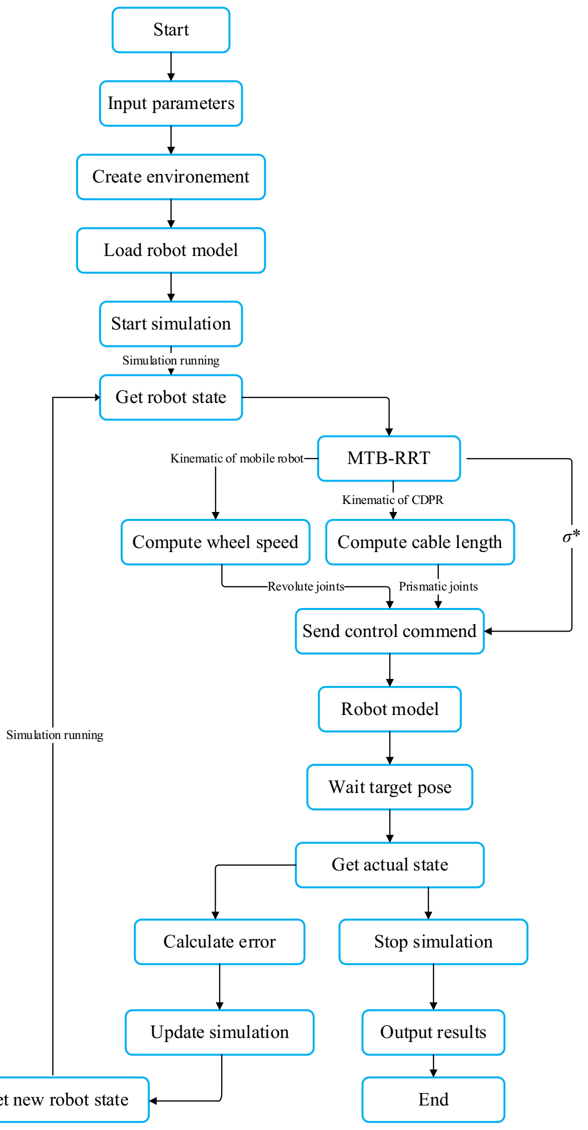


Fig. 11 Software simulation framework

This camera facilitated environment sensing, enabling the MCDPR to capture visual and depth information about its surroundings. Prior to conducting the experiment, the cables were pre-tensed to 20 N to ensure that cable tension remained positive during MCDPR movement. Moreover, the MCDPR incorporates two primary control systems. First, for the end-effector control, an inverse kinematic approach (equation 2) was employed to regulate its position accurately, which enables the determination of joint angles or cable lengths necessary to achieve the desired end-effector position, considering the kinematic constraints of the system. The control strategy for the mobile base systems involved the utilization of a differential drive control algorithm. This algorithm facilitates the tracking of a reference path by computing the required velocities of the left and right wheels of the robot.

Vision system

In this work, a vision system was designed to enable the MCDPR to accurately perceive its surroundings and determine its location. By leveraging the acquired self-state and environmental data, the proposed MTB-RRT algorithm consistently generates feasible paths for the MCDPR to navigate. To detect the obstacle, the YOLOv5 pre-trained model was employed, which can detect 80 different object categories [33]. YOLOv5 is a popular object detection algorithm that builds upon the success of the YOLO (You Only Look Once) family of models. It is a state-of-the-art real-time object detection system that achieves high accuracy and fast inference speeds. The YOLOv5 pre-trained model will output bounding boxes that indicate the location and size of each detected object. Moreover, the visual odometry simultaneous localization and mapping(OdoSLAM) was used [34]. In visual OdoSLAM, visual features such as corners or edges are detected and tracked over time to estimate the camera or robot's motion. By comparing the location of visual features between consecutive frames, visual OdoSLAM can estimate the relative pose of the camera with respect to the environment and thus the location of MCDPR can be realized.

Experimental results

The experimental results presented in Fig. 17 provide compelling evidence of the effectiveness of the MTB-RRT algorithm in acquiring accurate information about the robot's state and detecting the presence of obstacles. In the depicted scenario, as illustrated in Fig. 17a, the algorithm demonstrates its capability to identify the green static obstacle. Subsequently, by employing sampling and tree construction techniques in the state space, the algorithm successfully generates an optimal path that enables the robot to avoid the obstacle, as depicted in Fig. 17b. These experimental findings serve as a testament to the robustness and efficacy of the proposed algorithm in addressing obstacle avoidance challenges. The consistent acquisition of pertinent information and the ability to generate obstacle-free trajectories further validate the performance and potential of the MA-RRT algorithm.

Evaluation

Within this section, the comparative analysis of our proposed online motion planning method for MCDPRs against existing approaches was conducted. To ensure a fair evaluation, we specifically selected sampling-based online path planning methods for comparison. Our assessment encompassed three key aspects: path length, simulation time, and kinematic performance (J_e). To account for the stochastic nature of the proposed method, a comprehensive batch evaluation consist-

Fig. 12 Cable length change

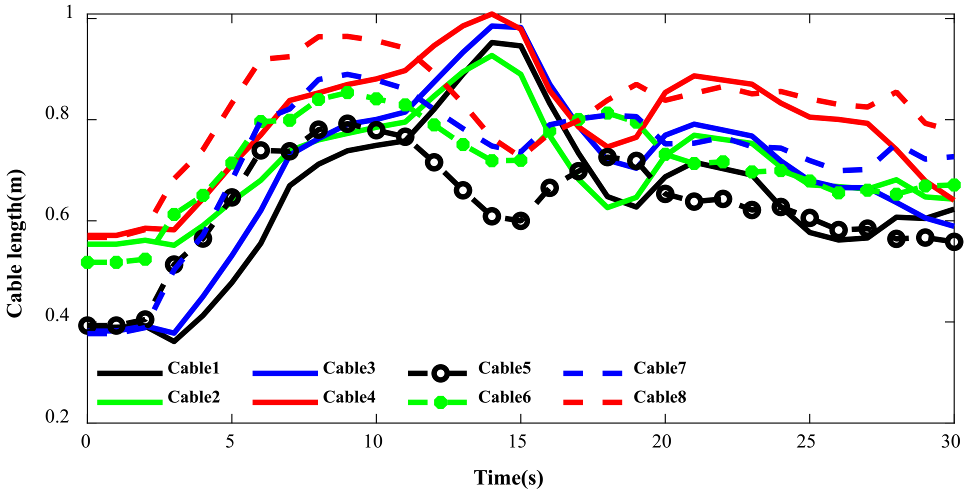
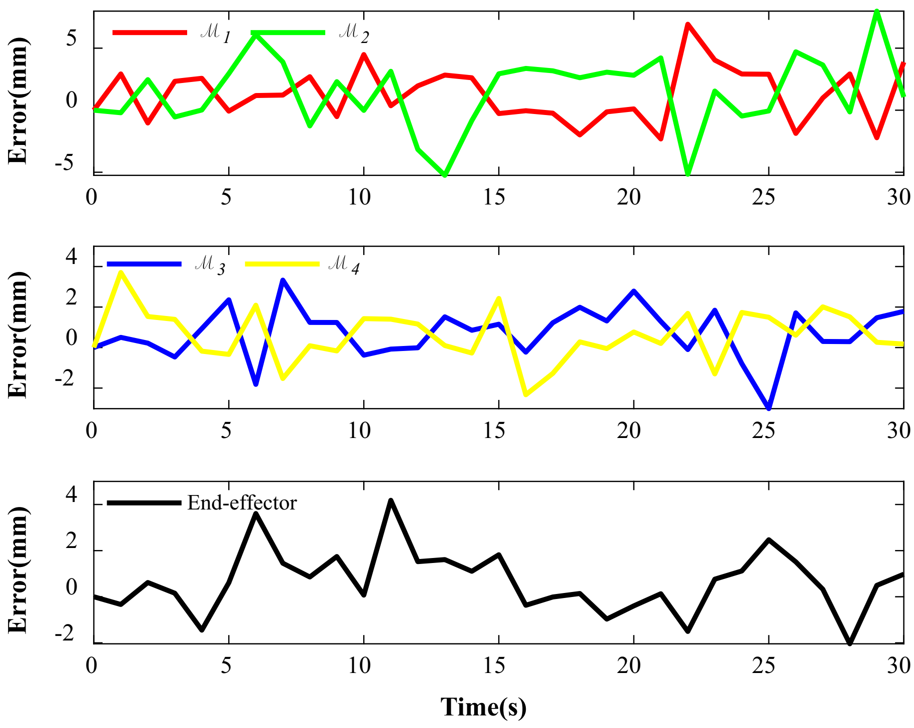


Fig. 13 Error between the actual and the desired position



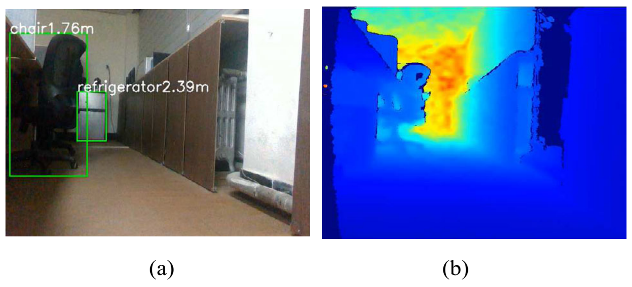


Fig. 14 Illstruation of YOLOv5

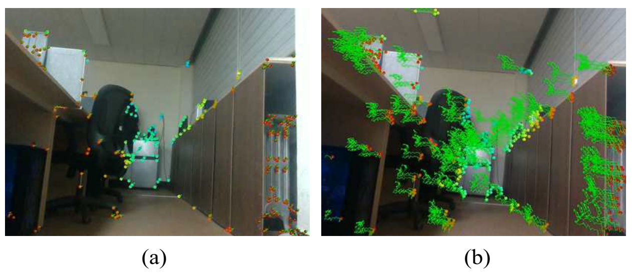


Fig. 15 Illstruation of OdoSLAM

ing of 1000 simulations was conducted. To ensure the realism and complexity of the scenarios, the environment was randomly generated for each simulation, carefully controlling the location, size, and number of obstacles. The maximum

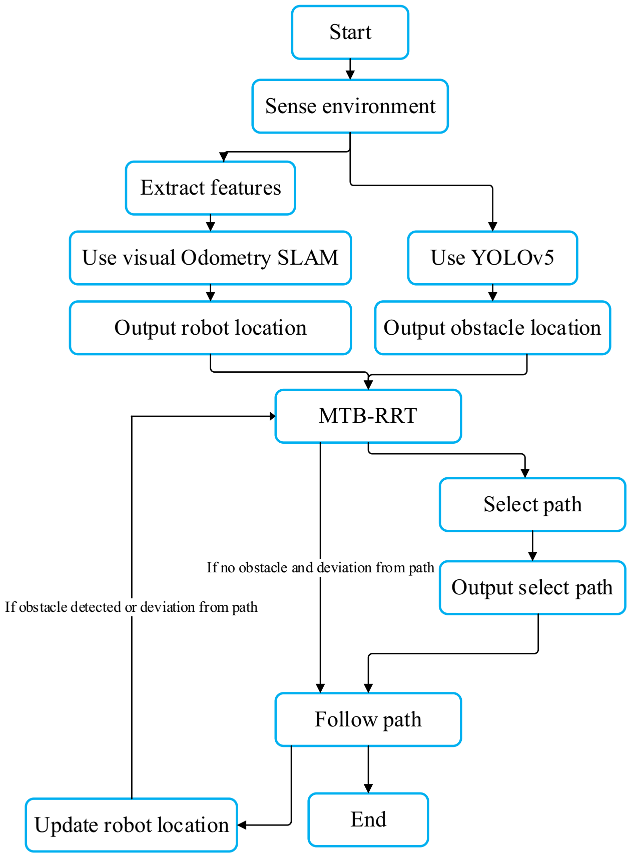


Fig. 16 Framework of vision system

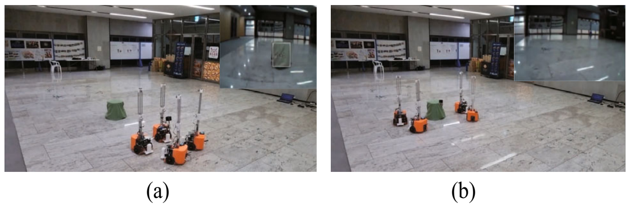


Fig. 17 Experimental results

number of obstacles was limited to nine, and their sizes were randomly assigned across three distinct scenarios. However, the starting and target points remained consistent throughout. This deliberate manipulation of obstacle characteristics aimed to create diverse and challenging environments, allowing for the evaluation of the proposed online motion planning approach's robustness and adaptability. By incorporating varying obstacle sizes and configurations, the study aimed to assess the method's performance in handling complex and unpredictable scenarios that resemble real-world conditions. The random generation of the environment provided a comprehensive assessment of the method's effectiveness, offering valuable insights into its capabilities for navigating through diverse and challenging environments encountered in MCDPR applications. Finally, the average values resulting from these simulations were used to evaluate the performance of the proposed method. All simulations were performed using MATLAB, utilizing the CPU computations of an Intel i7-9750 CPU @ 2.60 GHz with a 32 GB RAM Windows 10 system. To ensure consistency and comparability, the initialization parameters employed in the previous simulations were retained for these evaluations.

Table 2 The evaluation result

Method	CPU time	Path length	Average J_e
CL-RRT [15]	41.7 s	10.68 m	2.8
Reduction-based RRT* [16]	45.2 s	7.21 m	3.2
MRPT [19]	35.7 s	9.41 m	2.6
Adaptive RRT [20]	38.5 s	8.35 m	2.2
Proposed method	**30.5** s	**7.82** m	**4.1**

The results of the batch evaluation are summarized in Table 2. The proposed method outperforms the other methods with a significantly reduced CPU time, which can be primarily attributed to the introduction of the proposed cost function (Eq. 11). This improvement represents an impressive enhancement of approximately 18.6% when compared to the fastest competitor, MRPT. Furthermore, the proposed method exhibits a shorter path length with a reduction of 26.8% compared to CL-RRT. Although Reduction-based RRT* has the shortest path length, it is important to acknowledge that its optimization process detrimentally affects real-time performance, thereby limiting its practicality for online motion planning scenarios. Furthermore, since the proposed algorithm takes into account the kinematic performance of the robot (Eq. 15), the kinematic performance of the proposed method is the highest compared to other methods, especially compared to Adaptive RRT, with an improvement of 86.3%. Unlike another recent method that may rely on extensive training processes or data-driven models in [25], the proposed algorithm can be directly implemented without requiring such preparatory steps. These results serve as compelling evidence, validating the effectiveness and superiority of the proposed approach in the realm of online motion planning for MCDPRs.

Conclusion and future work

This paper presents an online motion planning method tailored for MCDPR systems, enabling them to generate feasible trajectories in unknown environments. The method involves the continuous generation of a time-based tree through a sampling of the state space as the MCDPR moves. The heuristic approach was employed in each iteration to determine the final path. By considering various constraints and employing designed cost functions, the proposed method ensures constraint satisfaction while maximizing system performance. Simulation in a complex environment using CoppeliaSim demonstrates that the technique creates practical trajectories for mobile bases and the end-effector within a shorter duration, resulting in enhanced kinematic performance and path quality. Empirical testing on a self-built MCDPR prototype, incorporating a vision system with YOLOv5 for environment perception and OdoSLAM for self-localization, validates the approach's validity. The simulation and experimental results substantiate the reliability of the proposed method.

Despite the promising performance of the proposed approach in real-world operations such as picking and releasing, there are several challenges that require further attention. Limited computational resources can impact execution time and overall performance. Environmental disturbances, including uneven terrain, dynamic obstacles, and varying lighting conditions, complicate trajectory planning. The parameter settings of the proposed method were based on experience and algorithm generality, necessitating further experiments and simulations to optimize performance. The assumption of prior knowledge of obstacle locations in simulations reduces viable trajectories in real-world scenarios. Imposed speed limits on mobile base and end-effector motions were necessary for localization accuracy, but exploring safe higher-speed operation remains a future research direction. Future work aims to enhance the intelligence and adaptability of MCDPR systems in navigating complex environments, contributing to their versatility and applicability in real-world operations.

Acknowledgements This study was supported by the National Research Foundation of Korea(NRF) grant funded by the Korean government. (MSIT) (2021R1A2C2013053).

Data Availability The data supporting the findings of this study are available within the article. Additionally, any additional data or materials related to this study can be requested from the corresponding author upon reasonable request.

Declarations

Conflicts of interest There is no conflict of interest.

References

1. Izard J-B, Dubor A, Hervé P-E, Cabay E, Culla D, Rodriguez M, Barrado M (2017) Large-scale 3d printing with cable-driven parallel robots. Construct Robot 1(1):69–76
2. Chen G, Baek S, Florez J.-D, Qian W, Leigh S.-w, Hutchinson S, Dellaert F (2022) Gtgraffiti: Spray painting graffiti art from human painting motions with a cable driven parallel robot. In: 2022 International Conference on Robotics and Automation (ICRA). IEEE, pp. 4065–4072
3. Chen Q, Zi B, Sun Z, Li Y, Xu Q (2019) Design and development of a new cable-driven parallel robot for waist rehabilitation. IEEE/ASME Trans Mechatron 24(4):1497–1507
4. Rasheed T, Long P, Marquez-Gamez D, Caro S (2019) Kinematic modeling and twist feasibility of mobile cable-driven parallel robots. In: Advances in Robot Kinematics 2018 16. Springer, pp. 410–418
5. Abbasnejad G, Eden J, Lau D (2018) Generalized ray-based lattice generation and graph representation of wrench-closure workspace for arbitrary cable-driven robots. IEEE Trans Robot 35(1):147–161
6. Gagliardini L, Gouttefarde M, Caro S (2018) Design of reconfigurable cable-driven parallel robots. In: Mechatronics for Cultural Heritage and Civil Engineering. Springer, pp. 85–113
7. Wang R, Li S, Li Y (2022) A suspended cable-driven parallel robot with articulated reconfigurable moving platform for schönflies motions. IEEE/ASME Trans Mechatron 27(6):5173–5184
8. Xiong H, Cao H, Zeng W, Huang J, Diao X, Lu W, Lou Y (2022) Real-time reconfiguration planning for the dynamic control of reconfigurable cable-driven parallel robots. J Mech Robot 14(6):060913
9. Pedemonte N, Rasheed T, Marquez-Gamez D, Long P, Hocquard É, Babin F, Fouché C, Caverot G, Girin A, Caro S (2020) Fastkit: a mobile cable-driven parallel robot for logistics. In: Advances in Robotics Research: From Lab to Market. Springer, pp. 141–163
10. Jaillet L, Cortés J, Siméon T (2010) Sampling-based path planning on configuration-space costmaps. IEEE Trans Robot 26(4):635–646
11. La Valle SM, et al (1998) Rapidly-exploring random trees: a new tool for path planning
12. Gammell JD, Srinivasa SS, Barfoot TD (2014) Informed rrt*: optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp. 2997–3004
13. Kuffner JJ, LaValle SM (2000) Rrt-connect: an efficient approach to single-query path planning. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), vol. 2. IEEE, pp. 995–1001
14. Spanogiannopoulos S, Zweiri Y, Seneviratne L (2022) Sampling-based non-holonomic path generation for self-driving cars. J Intell Robot Syst 104(1):14
15. Lin Y, Saripalli S (2017) Sampling-based path planning for uav collision avoidance. IEEE Trans Intell Transp Syst 18(11):3179–3192
16. Wen N, Zhang R, Wu J, Liu G (2020) Online planning for relative optimal and safe paths for usvs using a dual sampling domain reduction-based rrt* method. Int J Mach Learn Cybern 11:2665–2687
17. Zhang L, Lin Z, Wang J, He B (2020) Rapidly-exploring random trees multi-robot map exploration under optimization framework. Robot Auton Syst 131:103565
18. Lau BPL, Ong BJY, Loh LKY, Liu R, Yuen C, Soh GS, Tan U-X (2022) Multi-agv's temporal memory-based rrt exploration in unknown environment. IEEE Robot Autom Lett 7(4):9256–9263

19. Neto AA, Macharet DG, Campos MFM (2018) Multi-agent rapidly-exploring pseudo-random tree. J Intell Robot Syst 89:69–85
20. Xu J, Qian C, Park J-W, Park K-S (2022) Adaptive sampling-based moving obstacle avoidance for cable-driven parallel robots. IEEE/ASME Trans Mech 27(6):4983–4993
21. Xiang S, Gao H, Liu Z, Gosselin C (2020) Dynamic point-to-point trajectory planning for three degrees-of-freedom cable-suspended parallel robots using rapidly exploring random tree search. J Mech Robot 12(4)
22. Mishra UA, Mishra U, Métillon M, Caro S, et al (2021) Kinematic stability based afg-rrt* path planning for cable-driven parallel robots. In: The 2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
23. Rasheed T, Long P, Marquez-Gamez D, Caro S (2019) Path planning of a mobile cable-driven parallel robot in a constrained environment. In: International Conference on Cable-Driven Parallel Robots. Springer, pp. 257–268
24. Rasheed T, Long P, Roos A. S, .Caro S (2019) Optimization based trajectory planning of mobile cable-driven parallel robots. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 6788–6793
25. Liu Y, Cao Z, Xiong H, Du J, Cao H, Zhang L (2023) Dynamic obstacle avoidance for cable-driven parallel robots with mobile bases via sim-to-real reinforcement learning. IEEE Robot Autom Lett 8(3):1683–1690
26. Yan C, Xiang X, Wang C (2020) Towards real-time path planning through deep reinforcement learning for a uav in dynamic environments. J Intell Robot Syst 98:297–309
27. Chang W, Lizhen W, Chao Y, Zhichao W, Han L, Chao Y (2020) Coactive design of explainable agent-based task planning and deep reinforcement learning for human-uavs teamwork. Chin J Aeronaut 33(11):2930–2945
28. Nedic N, Stojanovic V, Djordjevic V (2015) Optimal control of hydraulically driven parallel robot platform based on firefly algorithm. Nonlinear Dyn 82:1457–1473
29. Nedic N, Prsic D, Dubonjic L, Stojanovic V, Djordjevic V (2014) Optimal cascade hydraulic control for a parallel robot platform by pso. Int J Adv Manuf Technol 72:1085–1098
30. Stojanovic V, Nedic N (2016) A nature inspired parameter tuning approach to cascade control for hydraulically driven parallel robot platform. J Optim Theory Appl 168(1):332–347
31. Tutsoy O, Barkana DE, Balikci K (2021) A novel exploration-exploitation-based adaptive law for intelligent model-free control approaches. IEEE Trans Cybern 53(1):329–337
32. Rohmer E, Singh SP, Freese M (2013) V-rep: a versatile and scalable robot simulation framework. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp. 1321–1326
33. Wu T-H, Wang T-W, Liu Y-Q (2021) Real-time vehicle and distance detection based on improved yolo v5 network. In: (2021) 3rd World Symposium on Artificial Intelligence (WSAI). IEEE 2021:24–28
34. Yousif K, Bab-Hadiashar A, Hoseinnezhad R (2015) An overview to visual odometry and visual slam: applications to mobile robotics. Intell Ind Syst 1(4):289–311