Intelligent Service Robotics (2019) 12:243–253
https://doi.org/10.1007/s11370-019-00278-7

ORIGINAL RESEARCH PAPER

Collision-free path planning of cable-driven parallel robots in cluttered environments

Jeong-Hyeon Bak[1] · Sung Wook Hwang[1] · Jonghyun Yoon[1] · Jong Hyeon Park[2] · Jong-Oh Park[3]

Received: 1 May 2018 / Accepted: 2 May 2019 / Published online: 27 May 2019

Abstract
Path planning of cable-driven parallel robots (CDPRs) is a challenging task due to cables which may cause various collisions. In this paper, three steps are suggested to perform path finding of CDPRs in cluttered environments. First, a way to visualize the cable collision of CDPRs is suggested to consider actual workspace of CDPRs. Second, a path finding algorithm based on rapidly exploring random trees (RRT) is presented to find a path free of various collisions of CDPRs including cable collisions and wrench feasible workspace. While conventional RRT algorithms are mainly focused on mazy environments, the modified RRT algorithm proposed here directly connects a sampled node and the tree to find a path faster in a non-mazy, but cluttered environment. Goal-biased sampling algorithm is also modified and employed to decrease computational cost. To deal with complicated collision detection of cables in the RRT, Gilbert–Johnson–Keerthi algorithm was employed. Finally, post-processing algorithm for any waypoint-based path is suggested to get a shorter and less winding path. A numerical study was carried out to suggest choosing proper meta parameters for the post-processing algorithm. The suggested algorithms were evaluated with 1000 times of simulations, and they were equally carried out for RRT* to compare. According to the results, the suggested algorithm found a shorter path with less computation time compared to RRT* and the post-processing algorithms made the already found path shorter.

Keywords Cable-driven parallel robot · Path planning · Cable collision · RRT

1 Introduction

Cable-driven parallel robots (CDPRs) are the ones actuated by winches parallelly connected to their end-effectors by cables. CDPRs have several advantages originated in a cable actuation. First, they have a vast variable workspace that makes it possible to utilize them in various environments. Additionally, CDPRs can manipulate heavy and large payloads with respect to their cost. That is because, unlike other link-based manipulators, actuators of a CDPR are generally fixed on the frame, not on a moving manipulator itself [1]. It reduces mass and inertia of manipulators, and allows the manipulator to handle relatively heavier things. For this reason, recently many researchers have had their research focus on CDPRs to employ their advantageous properties [2–5].

For using cable actuation in CDPRs, there are not only the advantages but also problems come from it. One issue focused in this paper is a problem of cable interferences in path planning of a CDPR in cluttered environments. Path planning is a classic issue in robotics, but still many researches on it are actively under way [6–9]. Because suitable path planning is needed for robots as they have different characteristics, which deeply affect the path planning. CDPR is one of the challenging platforms to plan a path because of cables. The cables of a CDPR are easily interfere with obstacles, end-effector, or themselves because they are hanging across the workspace. Cable interferences might cause an instantaneous tension drift on the cables, which would distort the entire trajectory of the end-effector. Consequently,

This research was supported by Leading Foreign Research Institute Recruitment Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (MSIP) (No. 2012K1A4A3026740).

✉ Jong Hyeon Park
jongpark@hanyang.ac.kr

1 Graduate School of Hanyang University, Seoul, Korea

2 Division of Mechanical Engineering, Hanyang University, Seoul, Korea

3 School of Mechanical Engineering, Chonnam National University, Gwangju, Korea

consideration of cable interferences is essential for path planning of CDPRs in cluttered environments.

Accordingly, researchers have studied for algorithms to detect interference of cables for CDPRs [10–12]. In [11], Blanchet et al. presented a detecting algorithm for cable interferences with environments and self-interferences. They considered cable volume and used a catenary model to consider cable sagging for more realistic collision detection algorithm. Nguyen et al. suggested several algorithms to detect cable collisions for general spatial CDPRs [10]. In this research, the authors studied about two cases of interferences: collisions between cables and cables, and interferences between cables and the end-effector. Wang et al. [12] presented cable collision-free area by clustering obstacles.

To operate CDPRs in cluttered environments, path planning is another important issue, because cable interference detection algorithms are not possible to prevent collisions. But there are not enough researches on path planning of CDPRs in cluttered environments. Lahour et al. proposed a path planning algorithm for CDPRs which has two behavior modes according to distances between end-effector and obstacles [13]. The authors used dual mode strategy based on a uniform grid to achieve the proposed obstacle avoiding path planning. It can find a path to goal if it exists. But the two core methodologies of it, which are dual mode and base of grid, have disadvantages. First, the suggested algorithm switches its mode from depth-first mode to width-first mode when the path meets obstacles. As a result, it generates paths which turn around obstacles. Those paths are clearly ineffective. Uniform grid-based approach also contributes to generation of ineffective path. Generally, grid-based path planning algorithm is fast enough to deal with real-time operation. But uniform grid-based path planning algorithm is a conservative obstacle avoiding algorithm because they have wasted space due to the grid size. Thus, there needs non-uniformly sampled grid-based path planning algorithm which will find more effective path.

Therefore, rapidly exploring random trees (RRT), which is a representative sampling-based path planning algorithm, is dealt with in this paper for more general and effective path planning of CDPRs in cluttered environments. RRT is a random sampling-based path planning algorithm suggested in [14]. It is possible to perform a path planning which considers complicated constraints and dynamics of a system with RRT [15]. Besides, it can be easily modified and applied to numerous system such as drones [16–18], mobile robots [19–21], autonomous vehicles [22,23], etc. This adaptability of RRT comes from properties of sampling-based planning (SBP). SBP randomly samples all possible configurations in C-space, and it guarantees that SBP finds an existing path if a number of sampling points is large enough. This characteristic is referred to as probabilistic completeness [15]. It is one of the essential properties of RRT, and any RRT-based planning algorithms should have it to assure that the algorithm is able to find a solution path if it exists. Recently, there exist several researches to reduce computation time by limiting sampling region [24,25].

In order to use RRT practically, a fast collision detection algorithm is necessary on account of needs for numerous sampling and collision checks in the sampling-based algorithms. There have been countless researches on collision detection algorithm for various usages [26–29]. Bergen [27], Klosowski et al. [28], and Gottschalk et al. [29] are very fast collision detection algorithms used for real-time collision detection, while they are less strict. These algorithms could not be considered for path planning of CDPRs because the workspace is much more complex and confined than what it looks like due to cables in the path planning of CDPR. An practical workspace, which considers cable collisions, can be visualized as suggested in Fig. 1 by perspective projection of cables' outlet points against the obstacles.

Therefore, fast and more accurate collision detection method is needed in using RRT for CDPRs. In this paper, Gilbert–Johnson–Keerthi (GJK) algorithm presented in [30] is used to detect various collisions. The algorithm is a kind of real-time collision detection algorithm based on Minkowski differentiation. One of the most advantageous properties of the algorithm is that it can strictly detect collisions between any shapes of convex polytopes [31]. Furthermore, it can calculate the minimum distance between two polytopes. These properties are appropriate to be used for path planning of CDPRs with RRT.

There already exists a research that presented about RRT-based path planning of CDPRs [32], but the research is only considering about a planar CDPR, and collisions between obstacles and cables are not considered. Considerations on the collision and spatial path planning are imperative in practical path planning of CDPRs. Hence, in this paper, we suggest a methodology to plan a path of CDPRs in

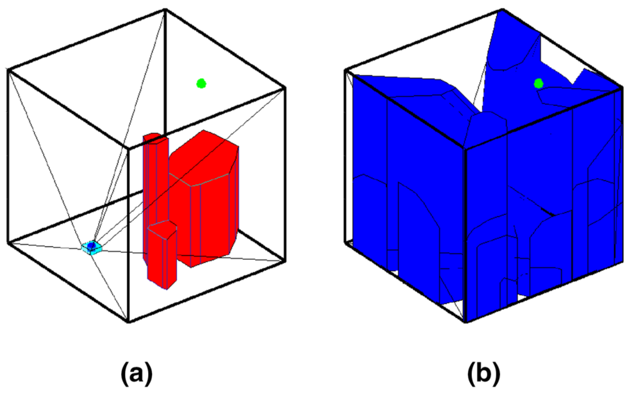

Fig. 1 **a** A CDPR with 3 random obstacles. **b** Visualization of obstacle shadow which is an virtual obstacles for cable anchored points on the end-effector

cluttered environments by using RRT and GJK algorithm. Interferences between cables, end-effector, and obstacles are considered to avoid undesirable collisions.

This paper is organized with 5 sections. The remaining sections are described as following. Problem definition and background knowledge are introduced in Sect. 2. In Sect. 3, we introduce our methodology to apply RRT for path planning of CDPRs in cluttered environments with considerations on various cable interferences, collisions, and wrench feasible workspace. To verify the suggestion, simulation and the result are presented in Sect. 4. Finally, the study is concluded in the last section.

2 Background

2.1 Considerations in path planning of CDPRs

Most of the problems in path planning of CDPRs are caused by cables because they are crossing the workspace by connecting end-effector and the fixed frame. Moreover, they only can hold positive tensions and have other nonlinearities such as sagging, hysteresis, and creep. Hence, path planning of CDPRs is a challenging task, especially in cluttered environments. In this paper, we present a methodology to perform a practical path planning for CDPRs based on sampling approach to consider these various issues of it.

Followings are some assumptions and considerations for the system and the algorithm suggested in this paper.

1. The cable is assumed to be a straight line for simplification of collision detection. It means that there is no cable sagging caused by any reason.
2. Outlets of cables, which generally mean the last pulleys of winches, are assumed to be fixed points. Outlet bias caused by cable-wrapping on pulleys is not considered. It also means that the last pulleys cannot be reconfigured.
3. Shapes of the end-effector and obstacles are assumed to be convex polyhedra. Concave polyhedron shape of the end-effector and obstacles can also be considered by separating them into a group of convex polyhedra. This means that the algorithm suggested in this paper can be applied to the almost whole shape of CDPRs by approximating them into polyhedra. That is possible with GJK algorithm suggested in [30,33].

Figure 2 shows the depicted CDPR model assumed in this paper.

2.2 RRT* algorithm

Rapidly exploring random trees star (RRT*) [34] is a well-known path finding algorithm based on the classic RRT [14].

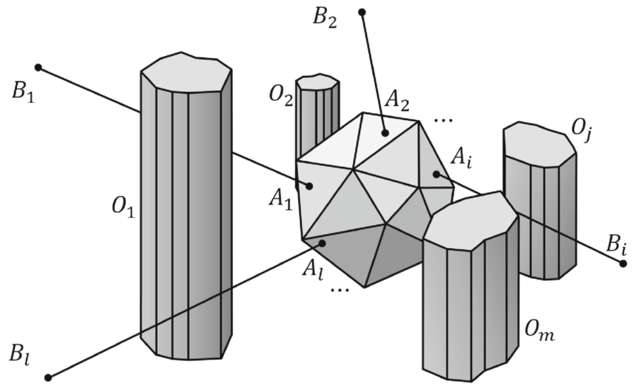

Fig. 2 In the model of the payload and the environment used in this paper, the end-effector is assumed to be a polyhedron supported by l number of straight cables from a fixed outlet B_i to an attaching point on the end-effector A_i. Outlet pulleys are fixed on the inertia frame and cannot be reconfigured. There exist m number of polyhedron-shaped obstacles O_j

RRT randomly searches all over the workspace, so it finds a winding path which is generally irrelevant to the optimal path. By this reason, RRT* investigates nodes nearby a new node and rewiring them by their costs. The pseudocode of the algorithm is shown in Algorithm 1.

Algorithm 1: RRT*

```
1  $T \leftarrow x_{\text{init}}$;
2  for $i$=1,...,$k$ do
3      $x_{\text{rand}} \leftarrow$RandomSample();
4      $x_{\text{nearest}} \leftarrow$Nearest($T, x_{\text{rand}}$);
5      $x_{\text{new}} \leftarrow$ Steer($x_{\text{rand}}, x_{\text{nearest}}$);
6      $flag \leftarrow$ CollisionFree($x_{\text{nearest}}, x_{\text{new}}$);
7      if $flag$ then
8          $X_{\text{near}} \leftarrow$ Near(T, $x_{\text{new}}$);
9          $x_{\text{parent}} \leftarrow$ FindParent($X_{\text{near}+}, x_{\text{nearest}}, x_{\text{new}}$);
10         $T \leftarrow$ Append($T, x_{\text{parent}}, x_{\text{new}}$);
11         $T \leftarrow$ Rewire($T, x_{\text{parent}}, X_{\text{near}}, x_{\text{new}}$);
12     end
13 end
14 return $T$
```

First, the algorithm randomly samples a state x_{rand} in a configuration space. Then, it finds the nearest node x_{nearest} in the tree T which will be a branch point for a new node. The algorithm finds a control input u that makes a state steer and drive toward the node x_{rand} from the nearest node within a unit time. For adding the new node to the tree, the new node is checked whether it collides with any obstacles during the steering process or not. If it does not, the algorithm computes costs and rewires every node nearby the new nodes.

As noted in the introduction, the algorithm is always possible to find a path to the goal if there exist enough number of sampling. The property is called ‘probabilistic completeness.’ Thanks to the property, the algorithm can be used to find a path in any cluttered or mazy environments. Furthermore, the algorithm is possible to consider various limitations and constraints with line 6 in Algorithm 1. For classic robots, such as serial manipulators and mobile robots, the step of the line only checks collisions of the robots themselves. But for CDPRs, it can consider various collisions and constraints such as cable collisions and wrench feasible workspace (WFW). Following subsections and sections describe considerations on those collisions and constraints.

2.3 GJK algorithm

Collision detection is one of the most important tasks in RRT* because it is the process that guarantees the generation of a collision-free path. Moreover, it greatly affects computation time of path planning algorithm. In this paper, random convex polyhedra-shaped obstacles and end-effector are assumed to consider more practical path planning. Therefore, fast and general collision detection algorithm is needed. GJK algorithm is a classic real-time collision detection algorithm for convex polytopes. At first, it is introduced as an algorithm for the shortest Euclidean distance computation algorithm between convex polytope set [30], but it is more famous as collision detection algorithm for convex polytopes. Nowadays, improved version of the algorithm [33,35] is widely used in many areas such as robotics, physics engine, and other geometric simulations. One limitation of the algorithm is a prerequisite that it can only be applied to convex polytopes. For this reason, concave obstacles and end-effector must be approximated as a convex polytope or decomposed into convex polytope subset.

Algorithm 2: GJK algorithm

```
k, $W_0$ ← Initialize();
for i=1,...,k do
    $w_i$ ← minNorm ($CH(W_{i-1})$);
    if $w_i == O$ then
        return true;
    else
        $W_i$ ← newSimplex($W_{i-1}, w_i$);
        $v_i \leftarrow s_{A\ominus B}(-w_i) = s_A(-w_i) - s_B(w_i)$;
        if $w_i == v_i$ then
            return false;
        end
    end
end
```

Fundamental pseudocode of the algorithm is shown in Algorithm 2 [35]. W_i is a simplex set which is the simplest convex polytope having $d + 1$ points in d dimension space. The function ‘minNorm’ finds a point which has the minimal norm in the given input space. ‘CH’ gives a convex hull set of the given points. ‘newSimplex’ gives a simplex points set which contains the given point and subset of the given simplex. $s_a(b)$ is called ‘support function’ which gives the extreme point along the b direction in a. $a \ominus b$ denotes the Minkowski difference which is defined as $\{x - y | x \in a, y \in b\}$. A and B are the target polytopes for collision check.

The algorithm iterates a sub-algorithm which makes a simplex in Minkowski difference try to contain the origin. Because, if the Minkowski difference contains the origin, the polytopes that make the difference collide with each other. Nowadays, the algorithm is working with a Voronoi diagram-based scheme to optimize the speed [35]. Hence, it is fast enough to check collisions in real time. This fast collision detection algorithm is proper to be used for RRT which must be iterated many times.

Furthermore, GJK algorithm is possible to detect collision between polytopes in different dimensions without any modification. This improves reusability and productivity of codes.

3 Considerations on limitations

Limitations in path planning of CDPRs must be considered in ‘CollisionFree’ function in Algorithm 1. In that function, feasibility of a path is checked including not only external collisions but also internal interference and getting out of WFW. The following subsections account for this.

3.1 External collision detection

External collision in this paper denotes collisions between obstacles and a part of CDPR such as cables and end-effector. The external collision is a fundamental consideration in most path planning problems. In RRT algorithm, collision detection must work during a robot’s step driving to a randomly sampled node. To detect a collision of a moving object, a swept volume obtained by sweeping the object along the moving trajectory is needed. If the swept volume is convex, its collision can be easily detected by applying GJK algorithm. But general moving trajectories are splines. Swept volume of it is generally concave and would not be separated into convex polytopes. Therefore, a linear driving scheme is used in RRT path planning for a simplification of the collision detecting problem. The convex swept volume with linear trajectory can be obtained by getting convex hull of points of the object at an initial point and a goal point. If the swept volume does not collide with any obstacles, collision-free movement of the object is guaranteed. Rotation of the end-effector is not considered for the same reason. Those exclusions can be considered with conservative assumptions, but it is not dealt with in this paper.

The difference between path planning of CDPR and that of other robots is the existence of cables. Not only colli-

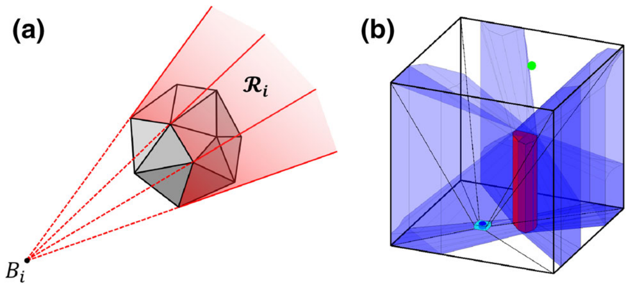

Fig. 3 **a** $\mathcal{R}_i$ denotes a region that a point A_i in Fig. 2 cannot pass through. The region is called 'obstacle shadow' in this paper. **b** Obstacle shadows for an obstacle in a fully constrained CDPR. The red polyhedron is the obstacle, and blue polyhedra are the obstacle shadows (color figure online)

sions between obstacles and end-effector, but also collisions between obstacles and cables must be considered in path planning of CDPR. To simplify the problem, cables are assumed to be straight and have no sagging. The points where cables are anchored on the end-effector are defined as A_i and the points where cables are connected to the fixed frame are defined as B_i in Fig. 2. With an assumption that cables do not sag, ith cable collides with an obstacle if A_i is behind the obstacle in the view of B_i. From this, cables–obstacles collision area can be depicted as like in Fig. 3a. By using this concept, cables–obstacles collision volume can be visualized as obstacles. Figure 3b depicts these volumes for a fully constrained CDPR. One difference between the volume and obstacle is that the ith volume only can affect specific point A_i, while obstacles affect the whole end-effector volume.

During a steering step of RRT, moving A_i and fixed B_i perform a triangular polygon in the space. To avoid cables interference, collisions between the triangular polygon and obstacles must be checked. GJK algorithm can detect collisions between polyhedron and polygon without modifying the algorithm.

3.2 Self-interferences

Basically, self-interference occurs due to cables which drive CDPRs. There may exist collisions between cables and cables, and cables and end-effector. Nguyen and Gouttefarde [10] and Makino and Harada [36] calculated distances between cables to find self-interferences of cables, but this method is not proper to plan a path and there can exist numerical errors.

The advantage of GJK algorithm is that it is possible to react any kind of collisions described above. As told in the external collision detection part, GJK algorithm can detect collisions between polytopes of a different dimension. Hence, it can also detect interferences between cables and cables, and cables and end-effector. An end of a cable is assumed to be fixed on the pulley; then, linear step driving of end-effector makes the other end move linearly. So the cable performs a triangle during the linear step driving of the end-effector. Collision detection between the triangles, and the triangles and end-effector sweeping volume using GJK algorithm allows preventing the self-interferences during the path planning of RRT.

3.3 Consideration of wrench feasible workspace

CDPRs have limitations due to cable properties because it is driven by cables. The most critical property that affects the CDPRs is the property that cables are possible to sustain only positive tension. Cables cannot exert reaction force against any negative tensions, namely compression. Hence, a configuration possible to generate negative tensile force must be avoided. Therefore, it is needed to express some wrench as a combination of positive tensile forces of cables. A configuration set which is possible to do it is defined as 'Wrench Feasible Workspace (WFW)' introduced in [37].

If a wrench set which can be generated by combinations of tensile forces is possible to contain a required wrench set, a configuration of the end-effector at that time is called 'wrench feasible' [37,38]. The wrench set is called 'available net wrench set,' and the required wrench set is called 'required net wrench set.' Available net wrench set is determined by configuration of the end-effector and minimum and maximum tensile force needed. Figure 4 describes a randomly configured CDPR (a) and force part of available net wrench set (b). Tensile forces are exerted along the cables, and they have minimum and maximum limit because of cable properties. By combining minimum and maximum tensile forces along the cables, raw available net wrench set can be acquired whose force part of it is expressed as a red convex polyhedron in Fig. 4b. In general, required force can be considered as a sphere at the force domain. Therefore, an inscribed sphere of the raw available net force set can be considered as effective available net force set. If the radius of the required force sphere is smaller than that of the effective available net force set, the configuration is force feasible. A moment feasibility

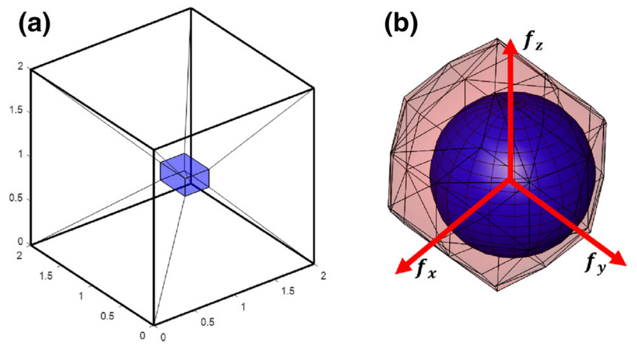

Fig. 4 **a** Random configuration of a CDPR. **b** Available force set

is possible to considered similarly. Finally, consideration of both feasibilities represents the wrench feasibility.

Then, a grid map of wrench feasibility against the configuration space can be obtained. Investigating position of the end-effector whether it is inside of the grid map or not is performed during the pathfinding algorithm. This will lead the generated path to stay in the wrench feasible workspace.

4 Modified RRT for CDPRs

The RRT* algorithm has several disadvantages to applying for CDPRs. First, a found path is too winding to track. It means that there exist a relatively large number of waypoints and they cause impact which generates oscillation on the end-effector. The oscillation can be avoided by generating smooth path or using input shaping introduced in [39]. A large number of waypoints have disadvantages in both cases. Furthermore, those winding paths have a large cost. Second, the fundamental RRT has relatively large calculation cost. It cannot reach the goal even with simple cases. This problem is caused because the fundamental RRT is developed for an uninformed pathfinding. But for path planning of CDPR in cluttered environments, the assumption of the uninformed situation is not necessary.

Two solutions are suggested in this study to solve the problems. Goal biasing [40] and direct connection to a randomly sampled point are the first solution, and post-processing of the generated path is suggested as the second solution. Modified goal-biased sampling is employed to reduce exploration time. Conventional goal-biased sampling algorithm or other tries to improve sampling like [40] samples based on a probability. It increases problem complexity by adding meta-variable of sampling probability. This is originated from step extend of conventional RRT. In this paper, direct connection to a sampled point is used instead of the conventional method. That method makes goal-biased sampling be deterministic. Hence, the concept of the goal-biased sampling is naturally merged with RRT. Algorithm 3 is presenting these process.

The algorithm is initialized and performs a goal sampling on demand. After every changes in tree T, next random sample is fixed on the goal point. For a not mazy environment, it dramatically decreases the number of iteration for the first reach to the goal. Then, 'FeasibleCheapest' finds the feasible and cheapest node on the Tree T. Path cost to find a cheap node is calculated as the sum of length from the root of the tree to a node and distance from the node to x_{rand}. Fundamental feasibility check is completely same with contents of Sect. 3, but there are some additional conditions for modified goal-biased sample. If x_{cheap} is equal to x_{goal} and path goal is already connected, x_{cheap} must be cheaper than the existing parent of x_{goal}. Otherwise, only conventional feasibility check is carried out to ensure the feasibility of x_{rand}. If there

Algorithm 3: Modified RRT algorithm

```
1  $T \leftarrow x_{\mathrm{init}}$;
2  $chkGoal \leftarrow true$;
3  for $i$=1,...,$k$ do
4      if $chkGoal$ then
5          $x_{\mathrm{rand}} \leftarrow x_{\mathrm{goal}}$;
6          $chkGoal \leftarrow false$;
7      else
8          $x_{\mathrm{rand}} \leftarrow$ RandomSample();
9      end
10     $x_{\mathrm{cheap}} \leftarrow$FeasibleCheapest($T$, $x_{\mathrm{rand}}$);
11     if $Exist(x_{\mathrm{cheap}})$ then
12         $T \leftarrow$ Append($T$, $x_{\mathrm{cheap}}$, $x_{\mathrm{rand}}$);
13         if $x_{\mathrm{rand}} == x_{\mathrm{goal}}$ then
14             Return $T$
15         else
16             $chkGoal \leftarrow true$;
17         end
18     end
19 end
20 return $T$
```

exist the feasible and cheapest node x_{cheap}, the algorithm connects it to x_{rand} and directly adds x_{rand} into the tree. This direct connection generates less winding paths and decreases searching time especially in simple cases. Additionally, if x_{rand} is equal to x_{goal}, the algorithm returns the tree T as the first-reached path. Unlike other RRT-based algorithms, the suggested algorithm only uses first-reached paths and does not perform further searching for a better paths because of a post-processing algorithm explained next.

The suggested modified RRT algorithm generates less winding path than conventional RRT algorithms. Still, resulting paths are expected to have relatively large cost. Hence, a post-processing algorithm is suggested to reduce the cost of resulting paths. The suggesting post-processing algorithm significantly reduces entire path cost, while it is a simple idea. Furthermore, it can be applied to any other path planning algorithm that consists of waypoints. The post-processing consists of two steps: a backward rewinding and forward rewinding. Algorithms 4 and 5 describe each steps of them.

There exist three for loops that are nested in the algorithms. The first for loop sweeps a reference node. The remaining nodes are explored by the second for loops. Then, 'CalculateStepVector' finds a step vector from the explored node to the next node of it in direction toward the reference node. This progress makes each rewinding algorithm try to connect the furthest waypoints if feasible, and the feasibility is checked with 'FeasibleToRewind.' Again, the word 'feasible' is described in Sect. 3. The algorithms choose a reference waypoint from the last or first, then choose backward or forward waypoints. The pair point x_{step} is a step forward or backward among the waypoints. The backward and forward rewinding algorithms are not individual but are sequential

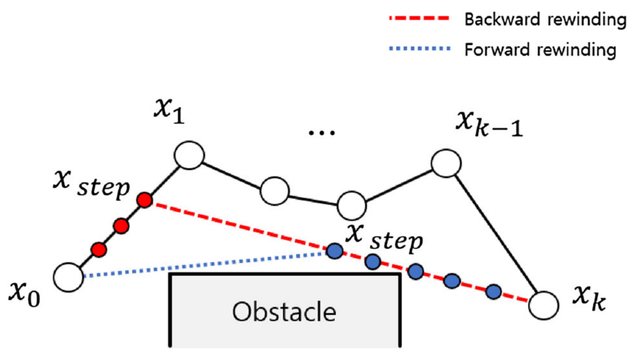

Fig. 5 Graphical expression of the proposed post-processing algorithm

algorithms. Effect of the sequence is analyzed in Sect. 5.2. Figure 5 delineates backward and forward rewinding algorithm. These algorithms can be conducted multiple times for more low-cost path, but cost-reducing effect compared to computational time radically decreases by repetition. This is also suggested in Sect. 5.2.

Algorithm 4: Backward Rewinding

```
for $i$=$n$,...,2 do
    for $j$=1,...,$i$-2 do
        step,m ← CalculateStepVector($x_j$, $x_{j+1}$);
        for $k$=0,...,$m$ do
            $x_{step} \leftarrow x_j + k * \text{step}$;
            if FeasibleToRewind($x_i$, $x_{step}$) then
                Rewind($x_i$, $x_{step}$);
                i←j+2;
                j←1;
                Break;
            end
        end
    end
end
```

Algorithm 5: Forward Rewinding

```
for $i$=0,...,$k$-2 do
    for $j$=$k$,...,$i$+2 do
        step,m ← CalculateStepVector($x_j$, $x_{j-1}$);
        for $k$=0,...,$m$ do
            $x_{step} \leftarrow x_j + k * \text{step}$;
            if FeasibleToRewind($x_i$, $x_{step}$) then
                Rewind($x_i$, $x_{step}$);
                k←UpdateWaypointNumber();
                j←k;
                Break;
            end
        end
    end
end
```

Table 1 Fundamental simulation parameters

Description	Spec.
Workspace dimension	2 m × 2 m × 2 m
End-effector dimension	0.15 m × 0.15 m × 0.05 m
Initial position	[0.3, 1, 0.3]
Goal position	[1.7, 1, 1.7]
Step distance	0.1 m
Obstacles height	0.5–1.5 m
Obstacles width	0.1–0.3 m

5 Simulation

In this section, several simulations are carried out to prove suggested path planning algorithms. They are conducted with commercial software MATLAB 2017b® on the operating system of 64-bit Windows 10 Pro. Intel® i5-4690 @ 3.5GHz CPU and 12GB DDR3 RAM is used for the computations. Fundamental simulation parameters are presented in Table 1. Prism-shaped obstacles are randomly generated all over the workspace. But those which collide with the initial and goal points are repeatedly regenerated until they do not collide with the points. Wrench feasible workspace, which is easily changed by the properties of cables and the end-effector, and the required wrench set, is ignored in this simulation for a convenience although it was considered in Sect. 3.3. An identical step distance was used for both RRT and the post-processing algorithms.

5.1 Path planning result

Figure 6 shows results of comparable path planning methods. Every path shown in the Figure does not collide with the obstacle in the workspace. RRT* and the suggested modified RRT algorithms are compared. Application of the post-processing algorithm is also compared. According to the results, suggested modification makes the resulting path have less number of waypoints. Furthermore, the post-processing algorithm makes both path planning algorithm result have less number of waypoints and shorter path. Further evaluation is presented at following subsections.

5.2 Post-processing

In this subsection, numerical studies on the suggested post-processing is performed. The study is progressed with parameters presented on Table 1. A simulation was carried out with 1000 times of iterations and obstacles are regenerated per each iteration. The simulation generates path using the modified RRT and then apply the post-processing algorithm for the generated path with step distance sweeping.

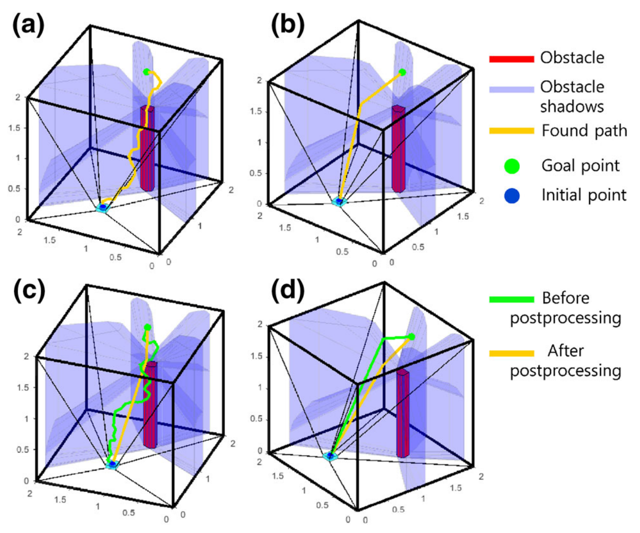

Fig. 6 Path planning result **a** RRT*. **b** Modified RRT. **c** RRT* + post-processing. **d** Modified RRT + post-processing

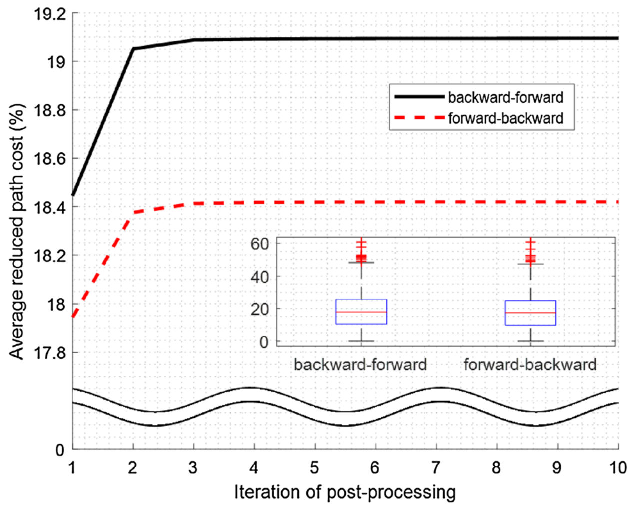

Fig. 8 Comparative simulation result between backward–forward and forward–backward post-processing; the box–whisker plot shows distribution of the first iteration for each method

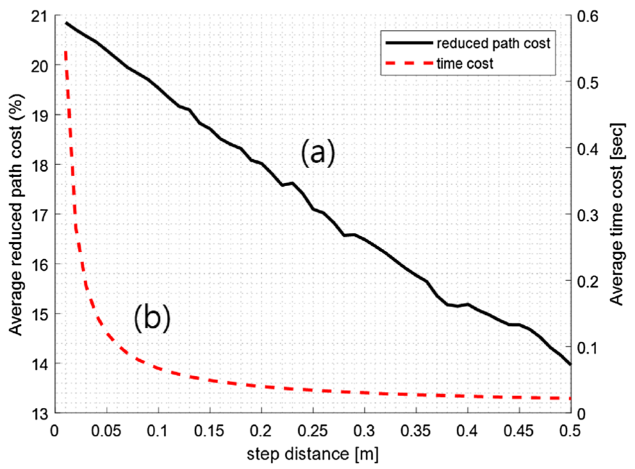

Fig. 7 Step distance sweeping result **a** average reduced path cost, **b** average computation time cost

The sweeping step distance starts from 0.01 (m) and end over 0.5 (m) with 0.01 (m) of step. Average results out of whole 1000 iterations are presented in Fig. 7. Figure 7a is a plot of average reduced path cost in percentage against step distance, and (b) is a plot of average computation time cost against step distance. Reducing step distance generates more effective path, while its computation time cost increases dramatically according to the results. Hence, step distance must be compromised between computation time and path cost by users.

In Fig. 8, analysis on sequence between backward and forward rewinding algorithms and their iterative property are depicted. Ten repeated iterations of post-processing were conducted with backward-first and forward-first method for 1000 times. Same waypoints are used for backward-first and forward-first post-processing algorithms. There averagely exist 0.66 %p of gap in their average cost reducing. And iterative use of the post-processing algorithm might reduce the path cost, but it is hard to expect huge path cost reduction. In conclusion, however, the sequence of two algorithms are still important because there are some outliers according to the result. Still there is no algorithmic pre-discrimination for the sequence so it is recommended to perform both methods and compare to chose a better one.

5.3 Batch evaluation

Batch evaluation is carried out to rate suggested algorithms numerically. Two kinds of environments illustrated in Fig. 9 are used to evaluate the suggested and conventional algorithms. Obstacles in the environments obstruct the way to goal much according to their shadows. Four algorithms noted in Fig. 6 are repeatedly conducted 1000 times per each. Batch evaluation results are presented in Table 2. Algorithms stop to find a path at their first reach to the goal.

According to the result, computing time of the modified RRT was only 0.87% of RRT*'s in the one-obstacle environment and 28.62% of RRT* in the three-obstacles environment. Its path cost was 19.88% shorter than RRT* in the one-obstacle environment and 13.40% shorter than RRT* in the three-obstacles environment. In the one-obstacle environment, the post-processing algorithm took 0.069 seconds for RRT* and 0.035 s for the modified RRT. In the three-obstacles environment, the post-processing algorithm took 0.241 seconds for RRT* and 0.126 s for the modified RRT

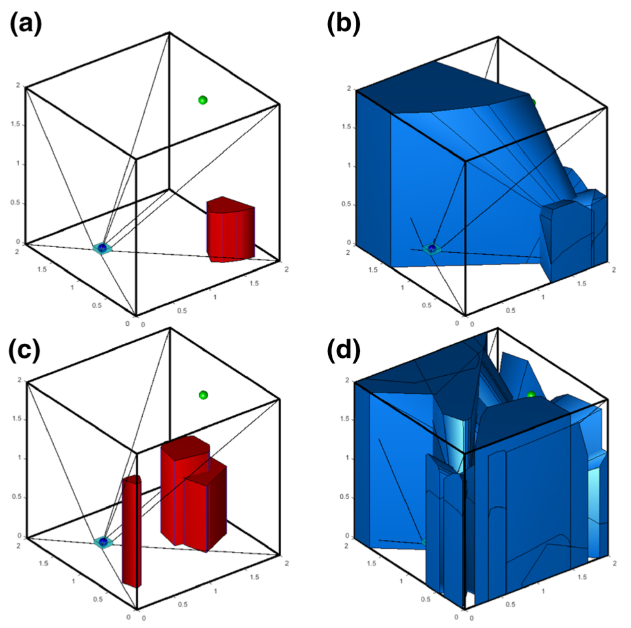

Fig. 9 **a** A one-obstacle environment. **b** Augmented obstacles due to the cables for the obstacle in (**a**). **c** A three-obstacle environment. **d** Augmented obstacles due to the cables for the obstacles in (**c**)

according to the calculation. It decreases average path cost by 35.31% for RRT* and 17.11% for modified RRT in the one-obstacle environment and 34.20% for RRT* and 18.44% for modified RRT in the three-obstacles environment. Note that standard deviations of computation time are extremely high due to randomness of RRT.

Furthermore, two-sample *t* tests were performed to signify difference between the data. According to *p* values of the group from A to R, results of the suggested algorithm are significantly different with conventional algorithms in spite of their large standard deviations originated from some outliers which can be observed in Figure 8. In contrast to this, the path calculation results without the post-processing are identical for group A, B, K and J. This is because their process is completely same except for the post-processing, and the pos-processing computing times and costs are noted apart. Especially note that calculation times in the group B might be looked different when their average value and standard deviations are the only focusing elements.

Computing time of RRT* does not change a lot according to the *t* test group A, B, K, and J even though the environment is extremely cluttered because of omnidirectional search of it. On the other hand, computing time of the proposed algorithm was strongly dependent on environmental complexity. Despite the dependency, computing time of the modified RRT was obviously shorter than that of RRT*. Path cost was minimal with RRT* and the post-processing in both cases. It is considered that relatively smaller step length of the RRT* may affect the post-processing.

Table 2 1000 times of batch evaluation results where A ∼ R denotes two-sample *t* test group

# of obstacles	Method	Average path calculation time (s)	SD of path calculation time (s)	Average post-processing time (s)	SD of post-processing time (s)	Average path cost (m)	SD of path cost (m)
1	RRT*	11.345 (B, C)	26.799	–	–	3.728 (G, H)	0.525
	RRT* + post-processing	10.567 (B, D)	22.756	0.069 (E)	0.015	2.412 (G, I)	0.087
	Modified RRT	0.099 (A, C)	0.166	–	–	2.987 (F, H)	0.338
	Modified RRT + post-processing	0.094 (A, D)	0.154	0.035 (E)	0.006	2.476 (F, I)	0.136
3	RRT*	10.962 (K, L)	12.540	–	–	4.710 (P, Q)	0.449
	RRT* + post-processing	10.831 (K, M)	12.345	0.241 (N)	0.040	3.099 (P, R)	0.153
	Modified RRT	3.138 (J, L)	4.076	–	–	4.079 (O, Q)	0.588
	Modified RRT + post-processing	3.320 (J, M)	4.029	0.126 (N)	0.022	3.327 (O, R)	0.275

**p* values of each groups—A: 0.5274, B: 0.4845, C–I < 0.001, J: 0.3148, K: 0.8137, L–R < 0.001

6 Conclusion

In this research, a methodology to find a path of CDPRs in cluttered environments without collision is suggested based on a modified RRT algorithm. A way to consider external and internal collision due to cables and wrench feasible workspace during path planning is suggested. Every collision was detected with GJK algorithm which is a fast multi-dimensional collision checking algorithm. Furthermore, a modified RRT and a post-processing algorithms were presented to find a less winding path for CDPRs. Suggested algorithms are evaluated with simulations. According to the result, the suggested algorithms found a path without any collision and the modified RRT algorithm found a shorter path with smaller computing time compared to RRT*. Also, the post-processing algorithm decreased path cost up to 35.25% in a simple environment. For the more complex environment, the post-processing algorithm decreased path cost up to 33.94%, while it only consumed 0.24 s of computing time out of 12.93 s to find the path.

References

1. Sohrabi S, Daniali HM, Fathi A (2016) Trajectory path planning of cable driven parallel manipulators, considering masses and flexibility of the cables. Int J Adv Des Manuf Technol 9(3):57–64
2. Pott A, Mütherich H, Kraus W, Schmidt V, Miermeister P, Verl A (2013) Ipanema: a family of cable-driven parallel robots for industrial applications. In: Cable-Driven Parallel Robots. Springer, pp 119–134
3. Gagliardini L, Caro S, Gouttefarde M, Wenger P, Girin A (2015) A reconfigurable cable-driven parallel robot for sandblasting and painting of large structures. In: Cable-Driven Parallel Robots. Springer, pp 275–291
4. Merlet JP (2017) Preliminaries of a new approach for the direct kinematics of suspended cable-driven parallel robot with deformable cables. In: New trends in mechanism and machine science. Springer, pp 355–362
5. Berti A, Merlet JP, Carricato M (2016) Solving the direct geometrico-static problem of underconstrained cable-driven parallel robots by interval analysis. Int J Robot Res 35(6):723–739
6. Mandava RK, Katla M, Vundavilli PR (2018) Application of hybrid fast marching method to determine the real-time path for the biped robot. Intell Serv Robot 12:1–12
7. Orozco-Rosas U, Montiel O, Sepúlveda R (2019) Mobile robot path planning using membrane evolutionary artificial potential field. Appl Soft Comput 77:236–251
8. Thabit S, Mohades A (2019) Multi-robot path planning based on multi-objective particle swarm optimization. IEEE Access 7:2138–2147
9. Taoudi A, Luo C (2019) Obstacle avoidance for a quadrotor using A* path planning and LQR-based trajectory tracking. In: AIAA Scitech 2019 Forum, p 1566
10. Nguyen DQ, Gouttefarde M (2015) On the improvement of cable collision detection algorithms. In: Cable-driven parallel robots. Springer, pp 29–40
11. Blanchet L, Merlet JP (2014) Interference detection for cable-driven parallel robots (CDPRs). In: 2014 IEEE/ASME international conference on advanced intelligent mechatronics (AIM). IEEE, pp 1413–1418
12. Wang B, Zi B, Qian S, Zhang D (2016) Collision free force closure workspace determination of reconfigurable planar cable driven parallel robot. In: Asia-Pacific conference on intelligent robot systems (ACIRS). IEEE, pp 26–30
13. Lahouar S, Ottaviano E, Zeghoul S, Romdhane L, Ceccarelli M (2009) Collision free path-planning for cable-driven parallel robots. Robot Auton Syst 57(11):1083–1093
14. LaValle SM (1998) Rapidly-exploring random trees: a new tool for path planning. Computer Science Dept Oct
15. LaValle SM, Kuffner JJ Jr (2000) Rapidly-exploring random trees: progress and prospects. In: Algorithmic and computational robotics: new directions. Citeseer
16. Dong Y, Fu C, Kayacan E (2016) RRT-based 3D path planning for formation landing of quadrotor UAVs. In: 14th international conference on control, automation, robotics and vision (ICARCV). IEEE, pp 1–6
17. Peng XZ, Lin HY, Dai JM (2016) Path planning and obstacle avoidance for vision guided quadrotor UAV navigation. In: 12th IEEE international conference on control and automation (ICCA). IEEE, pp 984–989
18. Lu L, Zong C, Lei X, Chen B, Zhao P (2017) Fixed-wing UAV path planning in a dynamic environment via dynamic RRT algorithm. In: Mechanism and machine science: Proceedings of ASIAN MMS 2016 & CCMMS 2016. Springer, pp 271–282
19. Palmieri L, Arras KO (2014) A novel RRT extend function for efficient and smooth mobile robot motion planning. In: IEEE/RSJ international conference on intelligent robots and systems (IROS 2014). IEEE, pp 205–211
20. Kamarry S, Molina L, Carvalho EÁN, Freire EO (2015) Compact RRT: a new approach for guided sampling applied to environment representation and path planning in mobile robotics. In: 12th Latin American Robotics Symposium (LARS) and 3rd Brazilian Symposium on Robotics (LARS-SBR). IEEE, pp 259–264
21. Moon C, Chung W (2016) Practical probabilistic trajectory planning scheme based on the rapidly-exploring random trees for two-wheeled mobile robots. Int J Precis Eng Manuf 17(5):591–596
22. Lan X, Di Cairano S (2015) Continuous curvature path planning for semi-autonomous vehicle maneuvers using RRT. In: European Control Conference (ECC). IEEE, pp 2360–2365
23. Kuwata Y, Fiore GA, Teo J, Frazzoli E, How JP (2008) Motion planning for urban driving using RRT. In: IEEE/RSJ international conference on intelligent robots and systems. IEEE, pp 1681–1686
24. Noreen I, Khan A, Ryu H, Doh NL, Habib Z (2018) Optimal path planning in cluttered environment using RRT*-AB. Intell Serv Robot 11(1):41–52
25. 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 (IROS 2014). IEEE, pp 2997–3004
26. Ericson C (2004) Real-time collision detection. CRC Press, Boca Raton
27. Bergen G (1997) Efficient collision detection of complex deformable models using AABB trees. J Graph Tools 2(4):1–13
28. Klosowski JT, Held M, Mitchell JS, Sowizral H, Zikan K (1998) Efficient collision detection using bounding volume hierarchies of k-dops. IEEE Trans Vis Comput Graph 4(1):21–36
29. Gottschalk S, Lin MC, Manocha D (1996) Obbtree: a hierarchical structure for rapid interference detection. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. ACM, pp 171–180

30. Gilbert EG, Johnson DW, Keerthi SS (1988) A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE J Robot Autom 4(2):193–203
31. Dyllong E, Luther W (2004) The GJK distance algorithm: an interval version for incremental motions. Numer Algorithms 37(1–4):127–136
32. XueMei J (2015) Rapidly-exploring random tree (RRT)-based path planning for cable-driven parallel robot considering interference between cables and platform. Ph.D. thesis, Chonnam University, Gwangju, Korea
33. Bergen GV (1999) A fast and robust GJK implementation for collision detection of convex objects. J Graph Tools 4(2):7–25
34. Karaman S, Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. Int J Robot Res 30(7):846–894
35. Ericson C (2004) The Gilbert-Johnson-Keerthi (GJK) algorithm. SIGGRAPH Presentation
36. Makino T, Harada T (2016) Cable collision avoidance of a pulley embedded cable-driven parallel robot by kinematic redundancy. In: Proceedings of the 4th international conference on control, mechatronics and automation. ACM, pp 117–120
37. Ebert-Uphoff I, Voglewede PA (2004) On the connections between cable-driven robots, parallel manipulators and grasping. In: 2004 IEEE international conference on robotics and automation (ICRA 2004), vol 5. IEEE, pp 4521–4526
38. Bosscher P, Riechel AT, Ebert-Uphoff I (2006) Wrench-feasible workspace generation for cable-driven robots. IEEE Trans Robot 22(5):890–902
39. Hwang SW, Bak JH, Yoon J, Park JH, Park JO (2016) Trajectory generation to suppress oscillations in under-constrained cable-driven parallel robots. J Mech Sci Technol 30(12):5689–5697
40. Urmson C, Simmons R (2003) Approaches for heuristically biasing RRT growth. In: 2003 IEEE/RSJ international conference on intelligent robots and systems (IROS 2003), vol 2. IEEE, pp 1178–1183