Continuous Collision Detection for a Robotic Arm Mounted on a Cable-Driven Parallel Robot

Diane Bury, Jean-Baptiste Izard, Marc Gouttefarde, Florent Lamiraux

Continuous Collision Detection for a Robotic Arm Mounted on a Cable-Driven Parallel Robot

Diane Bury[1,2], Jean-Baptiste Izard[1] , Marc Gouttefarde[3], Florent Lamiraux[2]

***Abstract*— A continuous collision checking method for a cable-driven parallel robot with an embarked robotic arm is proposed in this paper. The method aims at validating paths by checking for collisions between any pair of robot bodies (mobile platform, cables, and arm links). For a pair of bodies, an upper bound on their relative velocity and a lower bound on the distance between the bodies are computed and used to validate a portion of the path. These computations are done repeatedly until a collision is found or the path is validated. The method is integrated within the Humanoid Path Planner (HPP) software, tested with the cable-driven parallel robot CoGiRo, and compared to a discretized validation method.**

I. INTRODUCTION

This paper deals with continuous collision checking for a cable-driven parallel robot (CDPR) with a robotic arm mounted on it. The method developped in this paper is applied to the robot CoGiRo, shown in Fig. 1, a redundantly-actuated cable-suspended CDPR developed by LIRMM and Tecnalia. The problem consists in determining exactly whether an input path between two configurations of the robot (consisting of the CDPR and the arm) is collision-free, or if there exists a configuration along the path for which one of the robot body or cable is in collision with another robot body, cable or static part of the environment. Although for academic instances of path planning problems, discretized collision checking is usually enough, exact or continuous collision checking is very important when dealing with real robots in industrial settings. In some applications like path length optimization for instance [1], it is important that if a path has been validated, any sub-path be computed as valid using the same validation algorithm. This is not the case with discretized collision checking since the samples tested for collision may not be the same.

The developed method is integrated within the existing open-source software Humanoid Path Planner (HPP) [2] which includes sampling-based planning algorithms such as Probabilistic RoadMaps (PRM) and Rapidly-exploring Random Trees (RRT), and different optimization methods.

Several previous works on CDPRs dealt with the issues of cable-cable, cable-platform and cable-object collisions. The determination of the loci of cable collisions within a prescribed workspace is presented in [3], [4] in the case of a constant-orientation workspace and in [5], [6], [7] in the case of a 6D workspace. Fast heuristic approaches are proposed in [7] whereas certified calculations based on interval analysis are introduced in [5], [6]. Besides, an approximate determination of the volume swept by a cable when the mobile platform of a CDPR moves within a prescribed workspace is discussed in [8] and the geometric determination of cable-cylinder collision loci within the workspace of a CDPR is dealt with in [9]. In the latter, the cylinder is a fixed object located inside the CDPR workspace. Moreover, in [10], the collision-free printing workspace is calculated for fully-constrained CDPRs intended to print large-dimension objects in a sequence of horizontal layers. While all these methods determine various types of cable collision loci within a prescribed workspace, other previous works address the issue of checking cable collisions along a prescribed CDPR mobile platform path [5], [11], [12], which is also the purpose of the present paper. In [12], a classic discretized collision checking applied to CDPRs is presented. More advanced methods based on interval analysis, which can account for parameter uncertainties and round-off errors in numerical calculation, are introduced in [5], [11].

Fig. 1. Cable-driven parallel robot CoGiRo with a 7-DOF robotic arm.

The contribution of the present paper is a continuous cable collision checking method integrated within the open-source software HPP which can test collisions along a prescribed path of the mobile platform of a CDPR. Compared to [12], the proposed method allows continuous collision checking instead of discretized checking. The methods presented in [5], [11] to check cable collisions along a path can be classified as continuous checking method. Indeed, a proper use of interval analysis ensures that there does not exist any robot configuration along the path where a cable collision can occur, taking into account model parameter uncertainties and round-off errors. However, to the best of our knowledge,

[1]Tecnalia France, Bat 6 CSU, 950 rue Saint-Priest, 34090 Montpellier, France diane.bury@tecnalia.com, jeanbaptiste.izard@tecnalia.com

[2]LAAS-CNRS, University of Toulouse, Toulouse, France florent.lamiraux@laas.fr

[3]LIRMM, Université de Montpellier, CNRS, Montpellier, France marc.gouttefarde@lirmm.fr

these methods have not yet been applied to a problem case similar to ours. In this paper, we propose a method that takes into acount not only collisions between the cables, but also between the cables and all other bodies of the robot or the environment. Our method is also easily extended to other types of continuous validation of a path, for example cable tension validation. Moreover, its implementation in HPP makes it easy to use with any new robot, by providing the necessary files modeling the new robot. The contribution of the present paper can thus be used with any CDPR.

Besides, Schwarzer [13] proposes a method to check collision of a multi-arm robot along a continuous path composed of linear interpolations in the joint space. The method is based on the computation of upper bounds on the relative velocities – linear and angular velocities – of each body in the reference frame of the other bodies. Then, given the distance – or a lower bound on the distance – between two bodies at a given parameter value, the method computes a time interval over which no collision can occur between the bodies. This method is already implemented in HPP.

Cables are not solid bodies because they deform over time: they are subject to elongation and sagging. Even when assimilating the cable to a perfect cylinder attached at one end to the mobile platform and at the other end to a fixed exit point on the structure base, the cable length varies depending on the pose of the platform. Collisions between the cables and the mobile platform are possible and must be checked, but since the cables are attached to the platform, the lower bound on the distance is zero. For those reasons, [13] cannot be directly applied to a CDPR.

The method proposed in the present paper is inspired from [13] and adapted to CDPR cable collisions. The method takes as input

- A set of *collision elements* that are constituted by pairs of objects among robotic arm bodies, platform and cables of the CDPR, and static obstacles in the environment.
- A path which is a linear interpolation in the joint configuration space, called in this paper a *straight path*. The joint space of the CDPR consists of the 7 values for the pose of the mobile platform (3 values for the translation and a quaternion for the rotation) and of the values of the joints of the robotic arm.

In Section II, we present useful definitions and notations, and the overall algorithm used to validate a continuous path. Section III details the different methods used to calculate the velocity upper bound and the distance lower bound for the different types of pair of bodies. Implementation results in HPP are described in Section IV and Section V concludes the paper.

II. CONTINUOUS COLLISION CHECKING

A. Definitions and notations

We consider a CDPR composed of a mobile platform suspended by cables attached to a fixed frame structure, as depicted in Fig. 2. An articulated robotic arm is mounted on the platform for the purpose of grabing and moving objects in the environment. Some simplifications are made to modelize the cables:

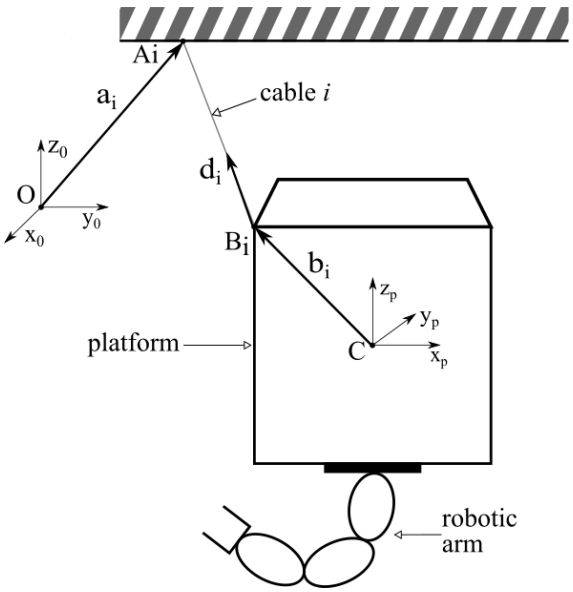

Fig. 2. Notations

- The cables are considered to be cylinders.
- Sagging of the cables is either neglected or taken into account in the cylinder radius.
- Each cable exits the fixed structure at a fixed point A_i.
- Each cable is attached to the mobile platform at a point B_i, fixed in the platform frame.

During motion planning, *straight paths* are computed and must be checked against collisions. We assume that along a *straight path*, the platform rotates or translates at a constant linear or angular velocity — respectively $\mathbf{v}_p$ and $\boldsymbol{\omega}_p$ — in the fixed reference frame. To validate a path in regard to collisions, every pair of bodies has to be checked for collisions. Those pairs are called collision elements, and can be of different types: a cable-cable collision, a cable-platform collision, a cable-arm collision, or robot-robot collision. In the software, collision element types are represented as derived classes of an abstract class, `CollisionElement`. The path to validate is set at the very start of the algorithm for all collision elements. This class has a method `validateElement`, which takes as input the time parameter value t corresponding to the configuration to validate. The output is a Boolean value indicating whether or not the configuration at the time parameter t is valid, in which case the method also returns a collision-free interval containing t, denoted as $interval$. The size of $interval$ is computed using an upper bound on the velocity V_{max} of one of the bodies of the pair relative to the other, and a lower bound on the distance between the two bodies D_{min}. The calculations for those bounds differ depending on the type of the collision element as detailed in section III.

Algorithm 1 Validation of a *straight path* using the dichotomy method

function VALIDATESTRAIGHTPATH($path$)
  $t \leftarrow 0$
  $validSubset \leftarrow \emptyset$
  $valid \leftarrow$ True
  while $valid$ is True and $validSubset \neq [0, T]$ **do**
    $success, validInterval \leftarrow$ VALIDATEALLCOLLISIONELEMENTS(t)
    if not $success$ **then**
      $valid \leftarrow$ False
    else
      $validSubset \leftarrow validInterval \cup validSubset$
    end if
    $t \leftarrow$ middle of first interval of $\overline{validSubset}$
  end while
  if not $valid$ **then**
    return first interval of $validSubset$
  else
    return $[0, T]$
  end if
end function

B. Continuous validation algorithm

1) Validation of a straight path: The input path to validate is a concatenation of linear interpolation paths that are called *straight paths*. Thus we need to successively validate each *straight path* using the function VALIDATESTRAIGHTPATH, until a collision is found or until all *straight paths* have been validated.

Let us denote by $[0, T]$ the interval of definition of the *straight path* to validate. If I is a subset of real numbers, we denote by $\overline{I}$ the complement of I in $[0, T]$. Algorithm 1 validates a path by creating a subset of validated intervals, $validSubset$. This subset is initialized with the empty set. The algorithm then loops until $validSubset$ is equal to $[0, T]$. t is initialized to 0. Function VALIDATEALLCOLLISIONELEMENTS is called with t and returns an interval containing t valid for all the collision elements. $validSubset$ is augmented with this newly validated interval (the new $validSubset$ is the union of the previous $validSubset$ and the newly validated interval). Then t receives the parameter value of the middle of the first non-tested interval. The algorithm then progresses until either $[0, T]$ is entirely validated, or a collision is found.

2) Finding a valid interval for all collision elements around a given configuration: As detailed in Algorithm 2, the function VALIDATEALLCOLLISIONELEMENTS takes as input a valid parameter value in $[0, T]$ and returns a valid interval containing the input parameter value. The function loops over all the collision elements, stored in a list $collisionElemList$. A local variable $validInterval$ is initialized to $]-\infty; +\infty[$ and represents the interval currently validated. For each $collisionElem$, an interval valid for this collision element only is calculated with VALIDATEELEMENT and the total valid interval $validInterval$ is trimmed by doing an intersection with this newly calculated interval.

3) Validating an interval for one collision element: The function VALIDATEELEMENT returns an interval around the given time parameter t for a collision element. Each collision element stores the set of intervals currently validated for a path. This variable is reset each time the algorithm starts working on a new path. Before computing a valid interval around t, it is first checked if the interval to validate has already been validated.

If not, an upper bound V_{max} of the velocity of one of the bodies relative to the other body of the pair is calculated, as well as a lower bound D_{min} of the distance at parameter t between the two bodies of the element (see next section for the calculation methods). If a collision is found during the calculation of D_{min}, VALIDATEELEMENT returns $false$ and reports the collision. If no collision is found, the value of D_{min} is a strictly positive real number. We define the output valid interval $newValidInterval$ in (1).

$$newValidInterval = [t - \frac{D_{min}}{V_{max}}, t + \frac{D_{min}}{V_{max}}] \quad (1)$$

III. CALCULATION OF THE VELOCITY AND DISTANCE BOUNDS

A. Collision between two bodies of the robot

As previously stated, for a pair of non-cable bodies of the robot (mobile platform, fixed structure, arm body), we calculate V_{max} using the method of [13]. D_{min} is computed using the FCL library [14] (BSD License). If the pair is composed of a link and its child link, then the collision checking is disabled, because the two bodies either touch each other by design, or cannot be in collision due to joint bounds.

B. Collision between a cable and the platform

Because each cable i is by design in contact with the platform at its attachment point, the method of [13] used for two bodies of the robot cannot be used as it stands. We choose to simply consider a shortened version of the cable that stops at a fixed distance d of the cable attachment point B_i on the platform. We calculate V_{max} an upper bound on the velocity of any point of the cable relative to the platform, and D_{min}, a lower bound on the distance between the shortened cable and the platform.

A previous iteration of our algorithm used a sampling-based model, computed off-line, to reduce the computing time, but at the expanse of the accuracy.

Calculation of a velocity upper bound

We consider a *straight path*, for which the linear and angular velocities of the platform are constant and known as stated in Section II. They are respectively noted $\mathbf{v}_p$ and $\boldsymbol{\omega}_p$, and we note $\omega_p = \|\boldsymbol{\omega}_p\|_2$ and $v_p = \|\mathbf{v}_p\|_2$. C is the origin of the platform frame. We want to compute an upper bound on the velocities of all the points of cable i in the platform frame.

Algorithm 2 Validation of a configuration by validating each interval element

function VALIDATEALLCOLLISIONELEMENTS(t)
  $validInterval \leftarrow]-\infty, \infty[$
  for each collision element $collisionElem$ in $collisionElemList$ **do**
    $(valid, newValidInterval, report) \leftarrow$ VALIDATEELEMENT($collisionElem$, t, $validInterval$)
    if not $valid$ **then** ▷ there is a collision for $config(t)$
      return (False, $validInterval, report$)
    else ▷ there is no collision for $config(t)$
      $validInterval \leftarrow validInterval \cap newValidInterval$
    end if
  end for
  return (True, $validInterval$) ▷ $validInterval$ is continuously valid for every collision element
end function

For cable i, we consider an orthogonal local frame $\mathscr{F}_i^B$ centered on B_i with its x-axis aligned with $\overrightarrow{B_iA_i}$ and directed toward A_i. Let P_i be the point on cable i at a fixed distance r of B_i. The relation between the coordinate vector P_i^p of P_i in the platform frame and its coordinate vector P_i^i in the local cable frame $\mathscr{F}_i$ is given by (2).

$$\begin{pmatrix} P_i^p \\ 1 \end{pmatrix} = M_{i/p} \begin{pmatrix} P_i^i \\ 1 \end{pmatrix} \tag{2}$$

$M_{i/p} = \begin{pmatrix} R_{i/p} & T_{i/p} \\ 0\ 0\ 0 & 1 \end{pmatrix}$ is the homogeneous matrix representing the position and orientation of the local cable frame in the platform frame. $R_{i/p} \in SO(3)$ is a rotation matrix. $T_{i/p} = \overrightarrow{CB_i} = \mathbf{b}_i \in \mathbb{R}^3$ is the position of the attachment point B_i in the platform frame. Since B_i is fixed in the platform frame, $T_{i/p}$ is constant. With $P_i^i = (r\ 0\ 0)^T$ being constant, by differentiating (2) with respect to time, we get

$$\begin{pmatrix} \dot{P}_i^p \\ 0 \end{pmatrix} = \begin{pmatrix} [\boldsymbol{\omega}_{i/p}]_\times R_{i/p} & 0 \\ 0\ 0\ 0 & 0 \end{pmatrix} \begin{pmatrix} P_i^i \\ 1 \end{pmatrix} \tag{3}$$

where $\dot{P}_i^p$ is the velocity of point P_i in the platform frame, $[\boldsymbol{\omega}_{i/p}]_\times$ is the antisymmetric matrix corresponding to the cross product with the cable frame angular velocity vector $\boldsymbol{\omega}_{i/p} \in \mathbb{R}^3$ in the platform frame, of norm $\omega_{i/p}$. This gives:

$$\dot{P}_i^p = [\boldsymbol{\omega}_{i/p}]_\times R_{i/p} P_i^i \tag{4}$$

Therefore, we can bound the norm:

$$||\dot{P}_i^p||_2 \leq \omega_{i/p} ||P_i^i||_2 \tag{5}$$

By composition of angular velocities, we have $\boldsymbol{\omega}_{i/p} = \boldsymbol{\omega}_{i/o} + \boldsymbol{\omega}_{o/p}$, and by bounding the norm: $\omega_{i/p} \leq \omega_{i/o} + \omega_p$, with $\boldsymbol{\omega}_{i/0}$ being the angular velocity of the cable joint frame i in the global reference frame.

Since B_i is fixed in the platform frame, which has known fixed linear and angular velocities $\mathbf{v}_p$ and $\boldsymbol{\omega}_p$, B_i is moving around A_i with a linear velocity of norm $v_{B_i} \leq v_p + \omega_p b_i$. Since $\|\overrightarrow{A_iB_i}\|_2 \geq L_{i,min}$, we get (6) and then (7).

$$\omega_{i/0} \leq \frac{v_p + \omega_p b_i}{L_{i,min}} \tag{6}$$

$$\omega_{i/p} \leq \omega_p + \frac{v_p + \omega_p b_i}{L_{i,min}} \tag{7}$$

From (5) and (7), and knowing we also have an upper bound on $\|P_i^i\|_2$ which is $L_{i,max}$, we get:

$$||\dot{P}_i^p||_2 \leq L_{i,max} \left(\frac{v_p + \omega_p b_i}{L_{i,min}} + \omega_p \right) = V_{max} \tag{8}$$

where b_i is a known constant value depending on the design of the CPDR, and $L_{i,min}$ and $L_{i,max}$ can be considered either as constant values depending on the shape of the workspace, or calculated for each *straight path*. Eq. (8) gives an upper bound on the velocity of point P_i in the platform reference frame along a *straight path*.

Calculation of a distance lower bound

The FCL library is used to compute a lower bound D_{min} on the distance between a cable i and the mobile platform. A STL file represents the 3D collision model of the platform, while the cable is represented by a cylinder. Since the cable and the platform are connected at point B_i, as stated above, we consider a shortened portion of the cylinder $A_i\tilde{B}_i$ where $B_i\tilde{B}_i = d\frac{\overrightarrow{B_iA_i}}{||\overrightarrow{B_iA_i}||}$. d is a fixed distance chosen so that for any configuration, if a point of $[B_i\tilde{B}_i]$ is in collision with the platform, then at least a point of $[\tilde{B}_iA_i]$ is also in collision with the platform. By design, D_{min} will never be greater than d, and it should be noticed that the closest point between the platform and the cable will generally be the (virtual) end of the cable $\tilde{B}$, i.e. $D_{min} = d$. Choosing d as large as possible reduces the number of iterations necessary to validate an interval, and thus the computing time.

C. Collision between two cables

If two cables have the same attachment points on the platform or the same exit points on the base structure, collision checking between them is disabled since these two cables cannot collide. For all other cable pairs, a velocity upper bound V_{max} and a distance lower bound D_{min} are calculated and used in (1).

Calculation of a velocity upper bound

We need to find an upper bound on the velocities of all the points of cable i in the local frame of cable j. Let P_i be

the point on cable i at a fixed distance r of B_i, with P_i^j its coordinate vector in the local frame of cable j. Similarly to the calculations in Section III-B, we can write:

$$\begin{pmatrix} P_i^j \\ 1 \end{pmatrix} = M_{i/j} \begin{pmatrix} P_i^i \\ 1 \end{pmatrix} \tag{9}$$

where $M_{i/j} = \begin{pmatrix} R_{i/j} & T_{i/j} \\ 0\ 0\ 0 & 1 \end{pmatrix}$ denotes the homogeneous matrix defining the position and orientation of the local frame of cable i with respect to the local frame of cable j. $\boldsymbol{\omega}_{i/j}$ is the rotation matrix of norm $\omega_{i/j}$ of the local frame of cable i relative to the local frame of cable j.

Differentiating with respect to time, and knowing that $P_i^i = (r\ 0\ 0)^T$ and $T_{i/j} = \overrightarrow{B_j B_i}$ are constant:

$$\begin{pmatrix} \dot{P}_i^j \\ 0 \end{pmatrix} = \begin{pmatrix} [\boldsymbol{\omega}_{i/j}]_\times R_{i/j} & 0 \\ 0\ 0\ 0 & 0 \end{pmatrix} \begin{pmatrix} P_i^i \\ 0 \end{pmatrix} \tag{10}$$

We then have $\dot{P}_i^j = [\boldsymbol{\omega}_{i/j}]_\times R_{i/j} P_i^i$. This gives $\|P_i^j\|_2 \leq \omega_{i/j} \|P_i^i\|_2$. We have $\boldsymbol{\omega}_{i/j} = \boldsymbol{\omega}_{i/0} - \boldsymbol{\omega}_{j/0}$. Using (6) and knowing we also have an upper bound on $\|P_i^i\|_2$ which is $L_{i,max}$,, we obtain:

$$\|\dot{P}_i^j\|_2 \leq L_{i,max} \left(\frac{v_p + \omega_p b_i}{L_{i,min}} + \frac{v_p + \omega_p b_j}{L_{j,min}} \right) = V_{max} \tag{11}$$

Calculation of a distance lower bound

We use the FCL library to compute a lower bound on the distance between two cables, represented by cylinders. The radius of the cylinder depend on the actual diameter of the cable, and can take into account a safety margin if desired. For two cables, the distance between their attachment points on the platform is a fixed upper bound on the distance between the cables and thus on D_{min}. It means that the proposed method is slower for a CDPR with cables attached close to one another on the platform than for a CDPR with cables attached far from one another. Indeed, if the cable attachment points are close, D_{min} is small and the algorithm can only validate small portions of the path.

D. Collision between a cable and the robotic arm

We consider a cable i and a body of the robotic arm, attached to joint J_a. Let $J_0 = J_p$, J_1, ..., $J_{m-1} = J_a$ the list of joints linking the platform joint J_p to J_a. J_1 is the joint attaching the robotic arm to the platform.

Calculation of a velocity upper bound

Let P_i be a point on cable i, and P_i^a its coordinate vector in the joint frame J_a.

$$\begin{pmatrix} P_i^a \\ 1 \end{pmatrix} = M_{j_{m-2}/j_{m-1}} M_{j_{m-3}/j_{m-2}} \dots M_{j_0/j_1} M_{i/p} \begin{pmatrix} P_i^i \\ 1 \end{pmatrix} \tag{12}$$

Where $M_{j_k/j_{k+1}} = \begin{pmatrix} R_{j_k/j_{k+1}} & T_{j_k/j_{k+1}} \\ 0\ 0\ 0 & 1 \end{pmatrix}$ is the homogeneous matrix representing the position of joint J_k in the reference frame of J_{k+1}.

By differentiating (12) with respect to time, we obtain

$$\begin{aligned}
\begin{pmatrix} \dot{P}_i^a \\ 0 \end{pmatrix} = & \begin{pmatrix} [\omega_{j_{m-2}/j_{m-1}}]_\times R_{j_{m-2}/j_{m-1}} & \mathbf{v}_{j_{m-2}/j_{m-1}} \\ 0\ 0\ 0 & 0 \end{pmatrix} \dots \\
& \qquad \dots M_{j_0/j_1} M_{i/p} \begin{pmatrix} P_i^i \\ 1 \end{pmatrix} \\
& + M_{j_{m-2}/j_{m-1}} \begin{pmatrix} [\omega_{j_{m-3}/j_{m-2}}]_\times R_{j_{m-3}/j_{m-2}} & \mathbf{v}_{j_{m-3}/j_{m-2}} \\ 0\ 0\ 0 & 0 \end{pmatrix} \dots \\
& \qquad \dots M_{j_0/j_1} M_{i/p} \begin{pmatrix} P_i^i \\ 1 \end{pmatrix} \\
& + \dots \\
& + M_{j_{m-2}/j_{m-1}} \dots M_{j_0/j_1} \begin{pmatrix} [\omega_{i/p}]_\times R_{i/p} & \mathbf{v}_{i/p} \\ 0\ 0\ 0 & 0 \end{pmatrix} \begin{pmatrix} P_i^i \\ 1 \end{pmatrix}
\end{aligned} \tag{13}$$

Then by bounding the norm using properties of rigid-body transformations:

$$\begin{aligned}
||\dot{P}_i^a||_2 \leq\ & v_{i/p} + \omega_{i/p} ||P_i^i||_2 \\
& + v_{j_0/j_1} (||P_i^i||_2 + ||T_{i/p}||_2) \\
& + v_{j_1/j_2} (||P_i^i||_2 + ||T_{i/p}||_2 + ||T_{j_0/j_1}||_2) \\
& \dots \\
& + v_{j_{m-2}/j_{m-1}} (||P_i^i||_2 + ||T_{i/p}||_2 + ... + ||T_{j_{m-3}/j_{m-2}}||_2)
\end{aligned} \tag{14}$$

We note D_k the cumulative length of joint J_k:

$$D_0 = 0 \qquad D_k = \sum_{t=0}^{k-1} ||T_{j_t/j_{t+1}}||_2 \quad \text{for } k \geq 1 \tag{15}$$

Since $L_{i,max}$ being an upper bound on $||P_i^i||_2$, the origin B_i of the local cable frame is fixed in the platform frame so $v_{i/p} = 0$, and using (7), we obtain an upper bound on the velocity of all points of cable i relative to the frame of joint J_a:

$$\begin{aligned}
||\dot{P}_i^a||_2 \leq\ V_{max} = & \left(\omega_p + \frac{v_p + \omega_p b_i}{L_{i,min}} \right) L_{i,max} + \\
& \sum_{k=0}^{m-2} \left(v_{j_k/j_{k+1}} (L_{i,max} + ||T_{i/p}||_2 + D_k) \right)
\end{aligned} \tag{16}$$

$||T_{i/p}||_2$ and all the $||T_{j_k/j_{k+1}}||_2$ are known by design of the robot. The same reasoning can be applied to validate collisions between a cable and an object of the environment.

Calculation of a distance lower bound

The FCL library is used once again to compute a lower bound on the distance between a cable, represented as a cylinder, and the body of the robotic arm, represented by a 3D collision model.

IV. IMPLEMENTATION RESULTS IN HPP

We have implemented the continuous collision checking method proposed in Sections II and III in the software HPP. The CoGiRo CDPR is simulated in HPP. The continuous method is compared to a discretized collision checking method, which uses a configuration validation method and a time step τ.

The computer used to run the software HPP is an "Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz" with 4,096 KB of cache memory and 16 GB of RAM.

This benchmark is performed on CoGiRo with the robotic arm. Random *straight paths* are generated by shooting random configurations in the configuration space, and keeping

τ (s)	True pos	True neg	New true pos	False pos	False neg
0.1	633	362	5	0	0
0.01	634	362	4	0	0
0.001	634	362	4	0	0

TABLE I

RESULTS FOR THE CONTINUOUS VALIDATION COMPARED TO THE DISCRETIZED VALIDATION WITH A TIME STEP τ FOR 1000 RANDOM *straight paths*

the first two valid configurations. Each *straight path* is validated using the continuous method and the discretized method with different time steps. Results are put into five categories:

- True positive: a collision was both detected by the discretized and the continuous methods.
- True negative: no collision were detected either by the discretized nor the continuous method.
- New true positive: a real collision was found by the continuous method, but was not detected by the discretized method.
- False positive: a collision was detected by the continuous method but is not an actual collision as determined by the configuration validation method.
- False negative: the continuous method failed to detect a collision that was found by the discretized method.

Results are resumed in Tables I and II with different time steps tested for the discretized method. The smaller the time step is, the bigger the average computation time is for the discretized method.

As shown in Table I, there is no false positive, thanks to the fact that the FCL library only returns zero as a lower bound of the distance between two objects if they are actually in collision. The continuous method is garanteed to find every collision, thus there is no false negative. The discretized method misses collisions that the continous method is able to find, even when the time step is reduced. With a time step of 0.001s, the discretized method has a longer computing time and fails to detect collisions. These results show the efficiency of the continuous method, and its usefulness in situations where no collision is tolerated.

V. CONCLUSIONS

This paper introduced a method which extends a continuous collision checking method for a multi-arm robot to a CDPR with a robotic arm mounted on its platform, to take into account collisions including the cables. The method has been integrated within the software HPP and tested in simulations on the robot CoGiRo. Results show that the method is effective and is able to find collisions which are undetected by a discretized method even with a small time step. Although our method is exact and can validate a path regarding collisions, it does not ensure the path is adapted to be performed on the physical robot. The algorithm could be improved using a cost-base planning method to maximize distance to obstacles. Future work will include applications on the physical robot CoGiRo to plan movements in a cluttered environment.

computing	Continuous			Discretized, τ = 0.1s		
time (s)	min	mean	max	min	mean	max
True positives	1.8e-04	0.095	1.1	2.6e-04	9.7e-03	0.052
True negatives	0.18	0.57	2.3	0.011	0.033	0.057
All paths	0.27			0.018		

computing	Discretized, τ = 0.01s			Discretized, τ = 0.001s		
time (s)	min	mean	max	min	mean	max
True positives	2.6e-04	0.095	0.50	2.6e-04	0.92	4.6
True negatives	0.11	0.33	0.59	1.0	3.3	6.7
All paths	0.18			1.8		

TABLE II

COMPUTING TIMES FOR 1000 RANDOM *straight paths*

REFERENCES

[1] M. Campana, F. Lamiraux, and J. P. Laumond, "A gradient-based path optimization method for motion planning," *Advanced Robotics*, vol. 30, no. 17-18, pp. 1126–1144, 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01301233

[2] J. Mirabel, S. Tonneau, P. Fernbach, A.-K. Seppälä, M. Campana, N. Mansard, and F. Lamiraux, "HPP: a new software for constrained motion planning," in *IEEE/RSJ Intelligent Robots and Systems*, October 2016.

[3] J.-P. Merlet, "Analysis of the influence of wires interference on the workspace of wire robots," in *Advances in Robot Kinematics*, J. Lenarčič and C. Galletti, Eds. Dordrecht, The Netherlands: Springer, 2004, pp. 211–218.

[4] S. Perreault, P. Cardou, C. Gosselin, and M. Otis, "Geometric determination of the interference-free constant-orientation workspace of parallel cable-driven mechanisms," *ASME Journal of Mechanisms and Robotics*, vol. 2, no. 3, 2010.

[5] L. Blanchet, "Contribution la modlisation de robots cbles pour leur commande et leur conception," Ph.D. dissertation, Université de Nice Sophia-Antipolis, 2015.

[6] L. Blanchet and J.-P. Merlet, "Interference detection for cable-driven parallel robots (cdprs)," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 14)*, 2014, pp. 1413–1418.

[7] D. Q. Nguyen and M. Gouttefarde, "On the improvement of cable collision detection algorithms," in *Cable-Driven Parallel Robots*, T. Bruckmann and A. Pott, Eds. Springer, 2014, pp. 29–40.

[8] A. Pott, "Determination of the cable span and cable deflection of cable-driven parallel robots," in *Cable-Driven Parallel Robots*, C. Gosselin, P. Cardou, T. Bruckmann, and A. Pott, Eds. Springer, 2017, pp. 106–116.

[9] A. Martin, S. Caro, and P. Cardon, "Geometric determination of the cable-cylinder interference regions in the workspace of a cable-driven parallel robot," in *Cable-Driven Parallel Robots*, C. Gosselin, P. Cardou, T. Bruckmann, and A. Pott, Eds. Springer, 2017, pp. 117–127.

[10] M. Fabritius, C. Martin, and A. Pott, "Calculation of the collision-free printing workspace for fully-constrained cable-driven parallel robots," in *Proc. ASME International Design Engineering Technical Conferences*, no. DETC2018-85961, Québec city, Québec, Canada, 2018.

[11] J.-P. Merlet and D. Daney, "Legs interference checking of parallel robots over a given workspace or trajectory," in *Proc. IEEE Int. Conf. Robotics and Automation*, Orlando, Florida, may 2006, pp. 757–762.

[12] S. Lahouar, E. Ottaviano, S. Zeghoul, L. Romdhane, and M. Ceccarelli, "Collision free path-planning for cable-driven parallel robots," *Robotics and Autonomous Systems*, vol. 57, pp. 1083–1093, 2009.

[13] F. Schwarzer, M. Saha, and J.-C. Latombe, "Exact collision checking of robot paths," in *Algorithmic Foundations of Robotics V, STAR 7*, J.-D. B. et al., Ed. Springer, 2004, pp. pp 25–41.

[14] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *IEEE International Conference on Robotics and Automation*, 2012.