A Versatile Tension Distribution Algorithm for n-DOF Parallel Robots Driven by $n+2$ Cables

Marc Gouttefarde, Johann Lamaury, Christopher Reichert, Tobias Bruckmann

A Versatile Tension Distribution Algorithm for n-DOF Parallel Robots Driven by $n+2$ Cables

Marc Gouttefarde, Johann Lamaury, Christopher Reichert, and Tobias Bruckmann

***Abstract*—Redundancy resolution of redundantly actuated cable-driven parallel robots (CDPRs) requires the computation of feasible and continuous cable tension distributions along a trajectory. This paper focuses on *n*-DOF CDPRs driven by *n*+2 cables since, for *n* = 6, these redundantly actuated CDPRs are relevant in many applications. The set of feasible cable tensions of *n*-DOF *n*+2-cable CDPRs is a two-dimensional convex polygon. An algorithm that determines the vertices of this polygon in a clockwise or counterclockwise order is first introduced. This algorithm is efficient and can deal with infeasibility. It is then pointed out that straightforward modifications of this algorithm allow the determination of various (optimal) cable tension distributions. A self-contained and versatile tension distribution algorithm is thereby obtained. Moreover, the worst-case maximum number of iterations of this algorithm is established. Based on this result, its computational cost is analyzed in detail, showing that the algorithm is efficient and real-time compatible even in the worst case. Finally, experiments on two 6-DOF 8-cable CDPR prototypes are reported.**

I. Introduction

CABLE-DRIVEN parallel robots (CDPRs) consist essentially of a mobile platform (the end-effector) driven by cables in a parallel topology, the cable lengths being controlled by means of winches. They possess several advantages such as a potentially very large workspace. Some previous works focused on crane-like applications, e.g. [1]–[3], haptic interfaces [4], [5], rehabilitation [6]–[8], and giant radio telescopes [9], [10].

The context of this paper is the real-time control of redundantly actuated CDPRs intended for industrial applications. Experiments on two redundantly actuated 6 degree-of-freedom (DOF) 8-cable CDPR prototypes, CABLAR [11] (Fig. 1) and COGIRO [12], [13] (Fig. 2), will be presented. On the one hand, CABLAR is an example where actuation redundancy is required because using more cables than DOFs is a well-known necessary condition to fully constrain a CDPR mobile platform (wrench-closure), e.g. see [14], [15]. On the other hand, for CDPRs in a crane-like configuration [1]–[3] (suspended CDPRs), actuation redundancy is not required but it can be advantageous, e.g., it allows COGIRO to have a large workspace-to-footprint ratio. This paper is dedicated to redundantly actuated CDPRs driven by $n+2$ cables where n denotes the number of mobile platform DOF. Most existing redundantly actuated CDPRs are driven by $n+1$ or $n+2$ cables [11], [13], [14], [16]–[22], probably because it represents a trade-off between the benefits of actuation redundancy and the CDPR complexity and cost. For 6-DOF CDPRs, driving the platform with $n+2=8$ cables rather than with $n+1=7$ cables leads generally to a larger and more homogeneous workspace. It also makes the integration into a workshop or a warehouse easier since symmetric cable arrangements in a cuboid supporting frame can be used. Hence, 6-DOF CDPRs driven by eight cables are relevant in many applications. This paper focuses on redundantly actuated CDPRs but, in some

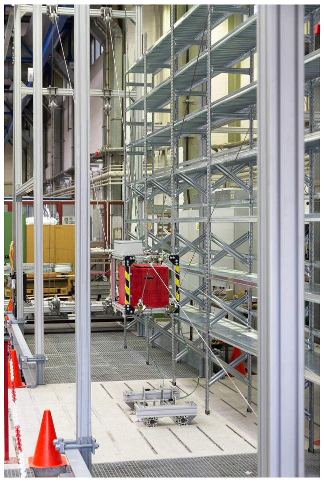


Fig. 1. CABLAR CDPR prototype. Source and Copyright ©: Lehrstuhl für Rechnereinsatz in der Konstruktion, University Duisburg-Essen.

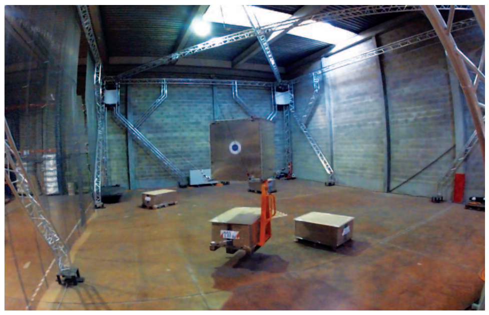


Fig. 2. LIRMM/TECNALIA COGIRO suspended CDPR prototype.

Marc Gouttefarde and Johann Lamaury are with the Robotics Department, Laboratory of Informatics, Robotics and Microelectronics of Montpellier (LIRMM, CNRS-UM), 161 rue Ada, 34095 Montpellier Cedex 5, France, e-mail: marc.gouttefarde@lirmm.fr.

Christopher Reichert and Tobias Bruckmann are with the Chair for Mechatronics, University of Duisburg-Essen, 47057 Duisburg, Germany.

The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement No. NMP2-SL-2011-285404 (CableBOT). The financial support of the ANR (grant 2009 SEGI 018 01) and of TECNALIA are also greatly acknowledged. The authors would like to thank the German Federal Ministry of Education and Research which supports the project "Entwicklung eines neuartigen Regalbediengerätes auf Basis der Stewart-Gough-Plattform" (01IC10L28A) within the "EffizienzCluster LogistikRuhr".

applications, CDPR driven by less than n cables, e.g. [23], may also be relevant.

The cable tension vector $\mathbf{t} \in \mathbb{R}^m$, m being the number of cables, is said feasible if it satisfies $\mathbf{0} \leq \mathbf{t}_{\text{min}} \leq \mathbf{t} \leq \mathbf{t}_{\text{max}}$. The tensions have to be non-negative since the cables cannot push on the mobile platform. The maximum values $\mathbf{t}_{\text{max}}$ are generally set by the breaking loads of mechanical parts or by the maximum actuator torques while positive values in $\mathbf{t}_{\text{min}}$ should avoid slack cables. In the case of redundantly actuated CDPRs, infinitely many *feasible cable tension distributions* exist when the platform pose (position and orientation) lies inside the wrench-feasible workspace (WFW) [24], [25]. For control purposes, the resulting redundancy resolution problem must be solved in real time and the computed cable tensions must be continuous along the prescribed mobile platform trajectory.

In several previous works, optimal tension distributions are computed. The objective function can be the 1-norm [2], [18], [19], [26] or ∞-norm [27] of $\mathbf{t}$ but these choices are prone to cable tension discontinuities along a trajectory [26], [28]. Hence, the p-norm should rather be used with $1 < p < \infty$ [29]. In particular, the 2-norm has often been selected, e.g. [30]–[34]. To compute such optimal tension distributions, iterative optimization algorithms can be used. Since the platform pose evolves continuously along a trajectory, warm start is possible which generally results in a reduced number of iterations before convergence, and thus in fast computations. Nevertheless, these iterative algorithms usually have an unknown, unbounded or very large worst-case computation time which is an issue for safe real-time implementations. This issue is solved in [34], [35] where the 2-norm and barycenter tension distributions are calculated with a relatively high but bounded worst-case complexity. Besides, the methods in [36]–[39] make simple, fast and predictable computations suitable for degrees of redundancy larger than two or three. However, these methods are not proved to work in the whole WFW [38] and they may not be effective for suspended CDPRs. To enlarge the part of the WFW in which they work, some cable tensions are set at their maximum or minimum admissible values, which may be contrary to the desired behavior and degrade the smoothness of the tension time evolutions (see the example in [39]). Furthermore, a general issue is the exact nature of the computed solution which cannot be chosen, e.g., no optimality criterion can be assigned to it.

Depending on the CDPR type (suspended or fully constrained) and on the required characteristics (e.g. high stiffness, low power consumption), the desired cable tension distribution is not always the same. For example, the minimization of energy consumption leads naturally to the 1-norm or 2-norm optimal tension distributions [2], [18], [19], [30]–[32] whereas the methods presented in [22], [26], [35] can be used to increase stiffness since they avoid low cable tensions. Therefore, a versatile method which can compute several types of cable tension distributions is of great interest. Moreover, the method should be mostly based on a single algorithm as independent as possible from specialized computational libraries (e.g. optimization algorithms) in order to avoid implementing or linking several different algorithms in a real-time control software environment. Additionally, a "safe" and reliable implementation requires a proved and real-time compatible worst-case computation time.

In most of the previously cited works, one type of tension distribution is computed. Exceptions are [22], [26] where parameters can be used to steer the cable tensions toward lower or higher values. However, such parameters may have a little effect as reported in [26] and the selection of their values may be an issue. Notably, they cannot be used to compute a prescribed tension distribution type (e.g. the 2-norm optimal distribution). Moreover, the method presented in [26] relies on a LP formulation so that discontinuities may occur and its worst-case complexity has not been established. In [22], the management of infeasibility is an issue and the use of an iterative optimization algorithm precludes a worst case complexity analysis. In [19], several tension distributions are computed but only low cable tension values are of interest, the computations mostly rely on optimization routines, and only planar CDPRs are considered. The method introduced in [39] could lead to versatile tension distribution computations but it has the drawbacks already pointed out above.

The present paper introduces a versatile tension distribution algorithm dedicated to n-DOF CDPRs driven by $n+2$ cables. For a given pose of the mobile platform of these CDPRs, the set of feasible cable tensions is known to be a convex polygon (Section II). The main contribution is an efficient geometric algorithm which either determines in order the vertices of this polygon or prove that it is empty (Section III). Moreover, the algorithm starting point is not required to correspond to a feasible cable tension vector. Another contribution is to point out that various (optimal) cable tension distributions can be obtained by straightforward uses or modifications of this algorithm (Section IV). Specifically, the determinations of the optimal 1-norm and 2-norm as well as the centroid and weighted barycenter tension distributions are explained. The worst-case maximum number of iterations of the algorithm is then established and its computational cost is analyzed (Section V). It is thereby proved to be efficient and well suited to real-time implementations. Its computational efficiency is also briefly compared to some previous methods. Finally, experimental results obtained on two 6-DOF 8-cable prototypes, the fully-constrained CDPR CABLAR and the suspended CDPR COGIRO, are presented (Section VI).

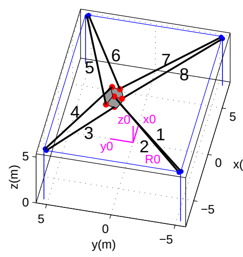
In the preliminary versions [13], [40] of the method introduced in this paper, the algorithm was not capable of dealing with infeasibility, its computational cost was not established, only the centroid cable tension distribution was computed, and experiments were conducted only on the COGIRO CDPR.

II. Feasible Cable Tension Polygon

The $n \times m$ *wrench matrix* $\mathbf{W}$ maps the cable tension vector $\mathbf{t} \in \mathbb{R}^m$ to the wrench $\mathbf{f} \in \mathbb{R}^n$ applied by the cables onto the mobile platform, e.g. [14], [15]

$$\mathbf{W}\mathbf{t} = \mathbf{f} \tag{1}$$

where m and n denote the number of cables and of mobile platform DOF, respectively. Let $r = m - n$ be the degree of


(a) Schematic view

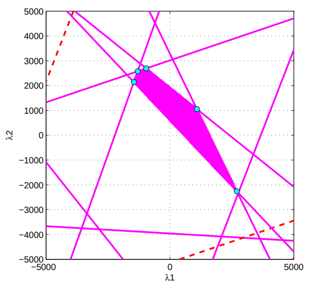

(b) Feasible polygon $\mathcal{A}^{-1}(\Lambda)$

Fig. 3. The CoGiRo suspended CDPR in static equilibrium at pose $[1\ 3\ 2.5\ 15\ 35\ 25]^T$ (units: meters and degrees, XYZ Euler angle convention) with a 300 kg platform mass, $\mathbf{t}_{\min} = 100$ N and $\mathbf{t}_{\max} = 5000$ N.

actuation redundancy (DOR). This paper is dedicated to the case $r = 2$ DOR. When $\mathbf{W}$ has full rank, Eq. (1) is equivalent to the following well-known equation

$$\mathbf{t} = \mathbf{W}^{+}\mathbf{f} + \mathbf{N}\boldsymbol{\lambda} \tag{2}$$

where $\mathbf{W}^{+}$ is the Moore-Penrose pseudoinverse of the wrench matrix, $\mathbf{N} = \text{null}(\mathbf{W})$ is a full rank $m \times 2$ matrix and $\boldsymbol{\lambda} = [\lambda_1\ \ \lambda_2]^T$ is an arbitrary 2-dimensional vector. The two columns of $\mathbf{N}$ form an orthonormal basis of the 2-dimensional nullspace of $\mathbf{W}$, i.e., $\mathbf{N}^T\mathbf{N} = \mathbb{I}_2$. $\mathbf{t}_p = \mathbf{W}^{+}\mathbf{f}$ is the minimum-norm solution of (1) satisfying $\mathbf{t}_p^T\mathbf{N} = \mathbf{0}$ and $\mathbf{N}\boldsymbol{\lambda}$ is the homogeneous solution where $\mathbf{N}$ maps $\boldsymbol{\lambda}$ into the nullspace of $\mathbf{W}$. Let $\Sigma \subset \mathbb{R}^m$ be the 2-dimensional affine space of the solutions to (1) and $\Omega \subset \mathbb{R}^m$ the m-dimensional hypercube of feasible cable tensions, i.e.,

$$\begin{aligned} \Sigma &= \{\mathbf{t} \mid \mathbf{W}\mathbf{t} = \mathbf{f}\} \\ \Omega &= \{\mathbf{t} \mid t_i \in [t_{\min}, t_{\max}],\ 1 \le i \le m\}. \end{aligned} \tag{3}$$

The intersection $\Lambda = \Omega \cap \Sigma$ is a 2-dimensional convex polytope representing the set of *feasible* cable tension distributions, e.g. [15], [28], [33]. Under the affine map $\mathcal{A} = (\mathbf{N}, \mathbf{t}_p)$, the preimage of Λ is also a 2-dimensional convex polytope, i.e., a convex polygon, defined as

$$\mathcal{A}^{-1}(\Lambda) = \{\boldsymbol{\lambda} \in \mathbb{R}^2 \mid \mathbf{t}_{\min} \le \mathbf{t}_p + \mathbf{N}\boldsymbol{\lambda} \le \mathbf{t}_{\max}\}. \tag{4}$$

In the sequel, $\mathcal{A}^{-1}(\Lambda)$ is refer to as the *feasible polygon*. For example, Fig. 3 shows the feasible polygon corresponding to a particular platform pose of a 6-DOF 8-cable suspended CDPR.

In fact, the feasible polygon $\mathcal{A}^{-1}(\Lambda)$ is defined by the following set of $2m$ linear inequalities

$$\mathbf{t}_{\min} - \mathbf{t}_p \le \mathbf{N}\boldsymbol{\lambda} \le \mathbf{t}_{\max} - \mathbf{t}_p \tag{5}$$

where each inequality defines a half-plane bounded by a line corresponding to values of $\boldsymbol{\lambda}$ for which one cable tension is equal to $t_{\min}$ or $t_{\max}$. The intersection of the $2m$ half-planes in (5) forms the feasible polygon $\mathcal{A}^{-1}(\Lambda)$.

The feasible polygon $\mathcal{A}^{-1}(\Lambda)$ has been considered only in a few previous works [19], [35]. It lies in the 2-dimensional space of vectors $\boldsymbol{\lambda}$ as opposed to Λ which is embedded into the m-dimensional space of cable tensions (in this paper, $m = 8$ for $n = 6$ DOF). Hence, the consideration of $\mathcal{A}^{-1}(\Lambda)$ allows to work in a 2-dimensional space where simple geometric reasoning can be made. In fact, Section III shows how polygon edges can be followed to reach a feasible point (if it exists) and then to compute all the feasible polygon vertices in order. The knowledge of the vertices (in a clockwise or counterclockwise order) completely determines the polygon geometry and thus allows a direct determination of various tension distributions (Section IV). The idea of following the polygon edges to compute its vertices is simple but, to the best of our knowledge, has never been used—at least in the context of CDPR cable tension distribution computation.

Note that, as pointed out in [19], [35], the vertices of $\mathcal{A}^{-1}(\Lambda)$ can be merely computed by solving all the 2×2 linear systems obtained by setting two of the inequalities in (5) to equalities. Each such system provides one vector $\boldsymbol{\lambda}$ which is an actual vertex of the polygon $\mathcal{A}^{-1}(\Lambda)$ if it verifies all the inequalities in (5). There are $2m$ inequalities in (5) and, when inequalities are set to equalities, each row of (5) defines two parallel straight lines. Hence, the total number of 2×2 linear systems to be solved is $C_2^{2m} - m = 4C_2^m$ [26] (112 in the case $m = 8$). This brute-force determination is simple but time-consuming. Moreover, the geometry of $\mathcal{A}^{-1}(\Lambda)$ is not fully revealed until the vertices are arranged in a clockwise or counterclockwise order. Hence, this brute-force determination must generally be followed by a convex hull and/or a triangulation algorithm [35] which further increases its computational cost. Section III will show that the vertices of $\mathcal{A}^{-1}(\Lambda)$ can be more efficiently computed in the case $m = n + 2$.

III. Efficient Determination of the Feasible Polygon Vertices

This section introduces the main contribution of this paper: An efficient algorithm which either determines the vertices of the feasible polygon or proves that the system of linear inequalities (5) is unfeasible. The algorithm can start at any intersection point (feasible or not) between two lines bounding half-planes defined by inequalities of (5). Moreover, the vertices are determined in a clockwise or counterclockwise order which allows a direct determination of various cable tension distributions (as discussed in Section IV).

A. Notations

Prior to the detailed description of the proposed algorithm, this section first introduces some notations.

The system of linear inequalities (5) is composed of m rows. Its row $i \in \{1, \ldots, m\}$ consists of two inequalities: $t_{min} - t_{p_i} \le \mathbf{n}_i\boldsymbol{\lambda}$ and $\mathbf{n}_i\boldsymbol{\lambda} \le t_{max} - t_{p_i}$, where the two-dimensional row vector $\mathbf{n}_i$ denotes the row i of $\mathbf{N}$. Each of these $2m$ inequalities defines a half-plane. In the sequel, the straight line bounding this half-plane is called an *inequality line* and the point of intersection between two of these inequality lines is referred to as an *intersection point*. Note that the two lines bounding the two half-planes defined by one row of (5) are parallel.

A row of (5) is said to be *satisfied* at a point $\boldsymbol{\lambda}$ if the two inequalities of this row are satisfied. A point is said to be *feasible* if all the rows of (5) are satisfied at this point, otherwise it is *unfeasible*. The *feasible index set* at a given point $\boldsymbol{\lambda}$ is the set consisting of the indices of the rows of (5) satisfied at $\boldsymbol{\lambda}$, e.g., if rows 2, 4, 5 and 7 are satisfied, the corresponding feasible index set is $\mathcal{I} = \{2, 4, 5, 7\}$. Moreover, the *feasible polygon $\mathcal{P}_\mathcal{I}$ associated to a given feasible index set* $\mathcal{I}$ is the set of points $\boldsymbol{\lambda}$ at which all the rows of (5) with indices in $\mathcal{I}$ are satisfied. $\mathcal{P}_\mathcal{I}$ is the intersection of the half-planes defined by the inequalities of these rows of (5). The edges and vertices of $\mathcal{P}_\mathcal{I}$ are segments of inequality lines and intersection points, respectively. For $\mathcal{I} = \{1, \ldots, m\}$, the feasible polygon associated to $\mathcal{I}$ is $\mathcal{P}_{\{1,\ldots,m\}} = \mathcal{A}^{-1}(\Lambda)$. The feasible index set associated to a feasible point is $\mathcal{I} = \{1, \ldots, m\}$.

B. Algorithm General Description

The main idea is to move along the inequality lines stopping at each intersection point encountered along the way. Each move from one intersection point to the next one must either keep the current feasible index set unchanged or, as often as possible, add one or several rows of (5) to this set. In other words, $\mathcal{I}$ being the feasible index set at some intersection point $\mathbf{v}$, a move along an inequality line to another intersection point $\mathbf{v}'$ is made only if the feasible index set $\mathcal{I}'$ at $\mathbf{v}'$ is such that $\mathcal{I} \subseteq \mathcal{I}'$. The number of rows of (5) which are satisfied at the current intersection point is thus equal to or greater than the number of rows satisfied at all the previously visited intersection points. In this way, the algorithm aims at reaching the feasible polygon $\mathcal{A}^{-1}(\Lambda)$, if it exists, and then at turning around it. In order not to retrace its steps, if the current intersection point $\mathbf{v}$ has been reached by following the inequality line L_i, the other inequality line intersecting L_i at $\mathbf{v}$ is followed in order to move to the next intersection point $\mathbf{v}'$.

Since there is a finite number of intersection points—a maximum of $4C_2^m$, see Section II,— the algorithm eventually reaches an intersection point which has already been visited before. Let $\mathbf{v}_f$ denote this intersection point, $\mathcal{I}_f$ be the feasible index set at $\mathbf{v}_f$ and $\mathcal{P}_{\mathcal{I}_f}$ the feasible polygon associated to $\mathcal{I}_f$. The intersection point $\mathbf{v}_f$ is a vertex of $\mathcal{P}_{\mathcal{I}_f}$ and the algorithm is terminated at $\mathbf{v}_f$. Indeed, as demonstrated in Section III-D, a full turn around $\mathcal{P}_{\mathcal{I}_f}$ has been made to go back to $\mathbf{v}_f$, i.e., all the vertices of $\mathcal{P}_{\mathcal{I}_f}$ were visited in a clockwise or counterclockwise order. If the algorithm were not stopped, the same full turn around $\mathcal{P}_{\mathcal{I}_f}$ would be made again and again.

Furthermore, only two cases are possible when the algorithm terminates at $\mathbf{v}_f$.

- *Case 1*: $\mathbf{v}_f$ is feasible, i.e., all the inequalities of (5) are satisfied at this point, $\mathcal{I}_f = \{1, \ldots, m\}$ and $\mathcal{P}_{\mathcal{I}_f} = \mathcal{A}^{-1}(\Lambda)$. In this first case, all the vertices of $\mathcal{A}^{-1}(\Lambda)$ have been visited and thus determined in a clockwise or counterclockwise order since the algorithm made a full turn around $\mathcal{A}^{-1}(\Lambda)$.
- *Case 2*: $\mathbf{v}_f$ is not feasible, $\mathcal{I}_f \neq \{1, \ldots, m\}$ and $\mathcal{P}_{\mathcal{I}_f} \neq \mathcal{A}^{-1}(\Lambda)$. In this second case, the inequality system (5) is proved to be unfeasible, i.e., $\mathcal{A}^{-1}(\Lambda) = \emptyset$. Otherwise, as demonstrated in Section III-D, the algorithm would have reached a vertex of $\mathcal{A}^{-1}(\Lambda)$.

Last but not least, because *feasibility is not required to start the algorithm*, the initial point $\mathbf{v}_{\text{init}}$ where the algorithm begins moving along the inequality lines can be the intersection point between any two inequality lines of (5).

C. Next Intersection Point

The determination of the next intersection point to be reached is the main part of the algorithm. It is repeatedly executed until the algorithm terminates.

Let us consider that the algorithm already reached point $\mathbf{v}_{ij}$ which is the intersection between the two inequality lines L_i and L_j. Index i (resp. j) designates the row of (5) containing the inequality which defines the halfplane bounded by L_i (resp. L_j). $\mathbf{v}_{ij}$ is not necessarily feasible and the feasible index set at $\mathbf{v}_{ij}$ is denoted $\mathcal{I}_{ij}$. Let us also consider that the algorithm reached $\mathbf{v}_{ij}$ by moving along L_j. Hence, line L_i must be followed in order to move to the next intersection point. The latter will be the *first* intersection point encountered along L_i while moving from $\mathbf{v}_{ij}$ in the direction which keeps the inequality corresponding to L_j satisfied. This inequality must be kept satisfied because index j is not authorized to leave the current feasible index set $\mathcal{I}_{ij}$.

First, let us determine the appropriate "direction of motion". Line L_i is to be followed and the row $\mathbf{n}_i$ of $\mathbf{N}$ is a vector orthogonal to L_i. Therefore, the appropriate direction of motion is a vector orthogonal to $\mathbf{n}_i$. Let this vector be the row vector $\mathbf{n}_{i_\perp}$: $\mathbf{n}_i \mathbf{n}_{i_\perp}^T = 0$. The points $\boldsymbol{\lambda}$ belonging to L_i are given by

$$\boldsymbol{\lambda} = \mathbf{v}_{ij} + \alpha \mathbf{n}_{i_\perp}^T \qquad (6)$$

where α is a scalar. Let us choose $\alpha \geq 0$ as the appropriate direction of motion along L_i. With the notation $\mathbf{n}_i = [a\ b]$, $\mathbf{n}_{i_\perp}$ can be equal to one of the following two vectors: $\mathbf{n}_{i_{\perp 1}} = [b \ \ -a]$ or $\mathbf{n}_{i_{\perp 2}} = [-b \ \ a]$. Between $\mathbf{n}_{i_{\perp 1}}$ and $\mathbf{n}_{i_{\perp 2}}$, the appropriate choice for $\mathbf{n}_{i_\perp}$ is the one such that the inequality corresponding to L_j remains satisfied for points $\boldsymbol{\lambda}$ given by (6). Then, $\mathbf{n}_j \boldsymbol{\lambda} = b_j - t_{p_j}$ being the equation of L_j, it is not difficult to prove that [40]:

- If $b_j = t_{min}$, among $\mathbf{n}_{i_{\perp 1}}$ and $\mathbf{n}_{i_{\perp 2}}$, the appropriate choice for $\mathbf{n}_{i_\perp}$ is the one such that $\mathbf{n}_j \mathbf{n}_{i_\perp}^T \geq 0$.
- If $b_j = t_{max}$, the appropriate choice for $\mathbf{n}_{i_\perp}$ is the one such that $\mathbf{n}_j \mathbf{n}_{i_\perp}^T \leq 0$.

The appropriate direction of motion $\mathbf{n}_{i_\perp}$ being known, the length of the move along L_i, i.e. the value of α in (6), which allows the algorithm to reach the next intersection point $\mathbf{v}_{li}$ must now be determined. This next intersection point corresponds to the smallest $\alpha \geq 0$ such that one of the inequalities of (5), apart from the two inequalities of row i, becomes an equality. Therefore, let us consider row k of (5)

$$t_{\text{min}} - t_{p_k} \leq \mathbf{n}_k \boldsymbol{\lambda} \leq t_{\text{max}} - t_{p_k}, \quad k \in \{1, \ldots, m\} \backslash \{i\} \quad (7)$$

where $\boldsymbol{\lambda}$ is given by (6). Let $L_{k,min}$ and $L_{k,max}$ be the inequality lines corresponding to the left-hand side and right-hand side inequalities of (7), respectively. The following cases have to be distinguished.

1. $\mathbf{n}_k \mathbf{n}_{i_\perp}^T = 0$: The two inequality lines $L_{k,min}$ and $L_{k,max}$ are parallel to L_i and no intersection point between L_i and these two lines can be found.
2. $\mathbf{n}_k \mathbf{n}_{i_\perp}^T > 0$:

a. If $\mathbf{n}_k\mathbf{v}_{ij} \le t_{\min} - t_{p_k}$, the intersection point between L_i and $L_{k,min}$ is obtained for $\alpha = \alpha_k = (t_{\min} - t_{p_k} - \mathbf{n}_k\mathbf{v}_{ij})/(\mathbf{n}_k\mathbf{n}_{i_\perp}^T) \ge 0$. L_i also intersects $L_{k,max}$ but for a larger value of α.

b. If $t_{\min} - t_{p_k} < \mathbf{n}_k\mathbf{v}_{ij} \le t_{\max} - t_{p_k}$, the intersection point between L_i and $L_{k,max}$ is obtained for $\alpha_k = (t_{\max} - t_{p_k} - \mathbf{n}_k\mathbf{v}_{ij})/(\mathbf{n}_k\mathbf{n}_{i_\perp}^T) \ge 0$. L_i also intersects $L_{k,min}$ but for a negative value of α.

c. If $t_{\max} - t_{p_k} < \mathbf{n}_k\mathbf{v}_{ij}$, there is no intersection $\boldsymbol{\lambda}$ between L_i and $L_{k,min}$ or $L_{k,max}$ such that $\alpha \ge 0$ in (6).

3. $\mathbf{n}_k\mathbf{n}_{i_\perp}^T < 0$:

a. If $\mathbf{n}_k\mathbf{v}_{ij} < t_{\min} - t_{p_k}$, there is no intersection point between L_i and $L_{k,min}$ or $L_{k,max}$ such that $\alpha \ge 0$.

b. If $t_{\min} - t_{p_k} \le \mathbf{n}_k\mathbf{v}_{ij} < t_{\max} - t_{p_k}$, $\alpha_k = (t_{\min} - t_{p_k} - \mathbf{n}_k\mathbf{v}_{ij})/(\mathbf{n}_k\mathbf{n}_{i_\perp}^T)$ is the nonnegative value of α such that L_i intersects $L_{k,min}$. L_i also intersects $L_{k,max}$ but for a negative value of α.

c. If $t_{\max} - t_{p_k} \le \mathbf{n}_k\mathbf{v}_{ij}$, the intersection point between L_i and $L_{k,max}$ is obtained for $\alpha_k = (t_{\max} - t_{p_k} - \mathbf{n}_k\mathbf{v}_{ij})/(\mathbf{n}_k\mathbf{n}_{i_\perp}^T) \ge 0$. It is smaller than the value of α for which L_i intersects $L_{k,min}$.

At most $m-1$ nonnegative scalars α_k are thereby computed. The smallest one

$$\alpha_l = \min_k \alpha_k \tag{8}$$

determines the next intersection point $\mathbf{v}_{li}$ where $l = \operatorname{argmin}_k(\alpha_k)$. The inequality line L_l which intersects L_i at $\mathbf{v}_{li}$ is either $L_{l,min}$ or $L_{l,max}$ depending on the sign of $\mathbf{n}_l\mathbf{n}_{i_\perp}^T$ and on the value of $\mathbf{n}_l\mathbf{v}_{ij}$ as detailed above.

The feasible index set $\mathcal{I}_{li}$ at $\mathbf{v}_{li}$ is such that $\mathcal{I}_{li} \supseteq \mathcal{I}_{ij}$. In fact, two cases should be distinguished. First, if $l \notin \mathcal{I}_{ij}$, $\mathcal{I}_{li} = \mathcal{I}_{ij} \cup \{l\}$ since row l of (5) is now satisfied (cases 2.a and 3.c) while all the rows of (5) with indices in $\mathcal{I}_{ij}$ remain satisfied because α_l has been chosen as the smallest α_k. A new feasible polygon $\mathcal{P}_{\mathcal{I}_{li}}$ has been reached and $\mathbf{v}_{li}$ is its first visited vertex. Second, if $l \in \mathcal{I}_{ij}$, we have $\mathcal{I}_{li} = \mathcal{I}_{ij}$ (cases 2.b and 3.b above). The algorithm kept on turning around the current feasible polygon $\mathcal{P}_{\mathcal{I}_{ij}} = \mathcal{P}_{\mathcal{I}_{li}}$ and:

- If $\mathbf{v}_{li}$ is not the first visited vertex of $\mathcal{P}_{\mathcal{I}_{ij}}$, the algorithm will keep on turning around this feasible polygon by looking for the next intersection point.
- Otherwise, the algorithm is back at the first visited vertex of the current feasible polygon $\mathcal{P}_{\mathcal{I}_{ij}} = \mathcal{P}_{\mathcal{I}_{li}}$. A full turn around this polygon has been made and the algorithm is terminated. If $\mathcal{I}_{li} = \{1, \ldots, m\}$, the inequality system (5) is feasible, the current feasible polygon is $\mathcal{A}^{-1}(\Lambda)$ and all the vertices of $\mathcal{A}^{-1}(\Lambda)$ have been determined. Otherwise, $\mathcal{I}_{li} \subset \{1, \ldots, m\}$ (strict inclusion) and the inequality system (5) is proved to be unfeasible ($\mathcal{A}^{-1}(\Lambda) = \emptyset$).

The flowchart of Fig. 4 summarizes the main steps of the algorithm introduced in this section.

D. Termination and Proofs

The algorithm presented in Sections III-B and III-C always goes back to an already visited intersection point, denoted $\mathbf{v}_f$, because there is a finite total number of intersection points (a maximum of $4C_2^m$). The proof below shows that, on the way back to $\mathbf{v}_f$, all the vertices of the feasible polygon $\mathcal{P}_{\mathcal{I}_f}$ are visited, where $\mathcal{I}_f$ denotes the feasible index set at $\mathbf{v}_f$.

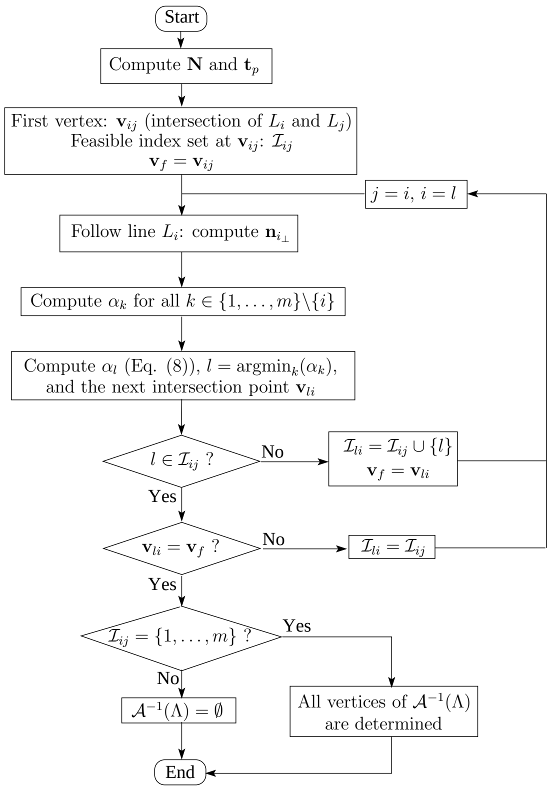

Start
Compute $\mathbf{N}$ and $\mathbf{t}_p$
First vertex: $\mathbf{v}_{ij}$ (intersection of L_i and L_j)
Feasible index set at $\mathbf{v}_{ij}$: $\mathcal{I}_{ij}$
$\mathbf{v}_f = \mathbf{v}_{ij}$
$j = i, i = l$
Follow line L_i: compute $\mathbf{n}_{i_\perp}$
Compute α_k for all $k \in \{1, \ldots, m\}\backslash\{i\}$
Compute α_l (Eq. (8)), $l = \operatorname{argmin}_k(\alpha_k)$, and the next intersection point $\mathbf{v}_{li}$
$l \in \mathcal{I}_{ij}$?
No
$\mathcal{I}_{li} = \mathcal{I}_{ij} \cup \{l\}$
$\mathbf{v}_f = \mathbf{v}_{li}$
Yes
$\mathbf{v}_{li} = \mathbf{v}_f$?
No
$\mathcal{I}_{li} = \mathcal{I}_{ij}$
Yes
$\mathcal{I}_{ij} = \{1, \ldots, m\}$?
Yes
No
$\mathcal{A}^{-1}(\Lambda) = \emptyset$
All vertices of $\mathcal{A}^{-1}(\Lambda)$ are determined
End

Fig. 4. Flowchart of the algorithm of Section III.

Proof 1. When the algorithm first reaches $\mathbf{v}_f$, it begins to turn around the convex polygon $\mathcal{P}_{\mathcal{I}_f}$, visiting the vertices of $\mathcal{P}_{\mathcal{I}_f}$ in a clockwise or counterclockwise order. Hence, assuming that one vertex of $\mathcal{P}_{\mathcal{I}_f}$ is not visited amounts to assume that a move from some vertex of $\mathcal{P}_{\mathcal{I}_f}$ led the algorithm to visit an intersection point $\mathbf{v}$ which is not a vertex of $\mathcal{P}_{\mathcal{I}_f}$. By construction of the algorithm (Section III-C), the feasible index set $\mathcal{I}$ at $\mathbf{v}$ satisfies $\mathcal{I}_f \subseteq \mathcal{I}$ since $\mathbf{v}_f$ was visited before $\mathbf{v}$. In fact, since $\mathbf{v}$ is not a vertex of $\mathcal{P}_{\mathcal{I}_f}$, we have $\mathcal{I}_f \subset \mathcal{I}$, i.e., $\mathcal{I}_f$ is a proper (strict) subset of $\mathcal{I}$. However, after having visited $\mathbf{v}$, the algorithm goes back to $\mathbf{v}_f$ so that, according to Section III-C, $\mathcal{I} \subseteq \mathcal{I}_f$. This is a contradiction since we have $\mathcal{I}_f \subset \mathcal{I}$ and $\mathcal{I} \subseteq \mathcal{I}_f$. Hence, $\mathbf{v}$ cannot be visited which proves that all the vertices of $\mathcal{P}_{\mathcal{I}_f}$ are necessarily visited since the algorithm cannot take another path on its way back to $\mathbf{v}_f$. □

This proof also demonstrates that if the algorithm were not stopped at $\mathbf{v}_f$, the same full turn around $\mathcal{P}_{\mathcal{I}_f}$ would be made repeatedly. Therefore, the algorithm is stopped at $\mathbf{v}_f$ and two cases are possible. First, if $\mathcal{I}_f = \{1, \ldots, m\}$, $\mathcal{P}_{\mathcal{I}_f} = \mathcal{A}^{-1}(\Lambda)$, the inequality system (5) is feasible and all the vertices of $\mathcal{A}^{-1}(\Lambda)$ have been determined in order. Second, if $\mathcal{I}_f \neq \{1, \ldots, m\}$, i.e. $\mathcal{I}_f \subset \{1, \ldots, m\}$, let us prove that the inequality system (5) is unfeasible, i.e., $\mathcal{A}^{-1}(\Lambda) = \emptyset$.

Proof 2. Assume, to the contrary, that $\mathcal{A}^{-1}(\Lambda)$ is not the empty set. Since $\mathcal{A}^{-1}(\Lambda) = \mathcal{P}_{\{1,\ldots,m\}}$ and $\mathcal{I}_f \subset \{1, \ldots, m\}$, $\mathcal{A}^{-1}(\Lambda)$ is included into $\mathcal{P}_{\mathcal{I}_f}$. Since $\mathcal{I}_f$ is a proper subset of $\{1,\ldots,m\}$ ($\mathcal{I}_f \subset \{1,\ldots,m\}$), there is $k \in \{1,\ldots,m\}$ such that $k \notin \mathcal{I}_f$. One of the two inequality lines associated to row k of (5), denoted L_k, is thus supporting an edge of $\mathcal{A}^{-1}(\Lambda)$ while cutting $\mathcal{P}_{\mathcal{I}_f}$ in two parts. Consequently, L_k intersects two edges of $\mathcal{P}_{\mathcal{I}_f}$ and the feasible index set at the two corresponding intersection points is $\mathcal{I}_f \cup \{k\}$. Since $\mathcal{I}_f \cup \{k\}$ is larger than $\mathcal{I}_f$, the algorithm necessarily stops at one of these two intersection points while turning around $\mathcal{P}_{\mathcal{I}_f}$. From this point on, it leaves the boundary of $\mathcal{P}_{\mathcal{I}_f}$ by following L_k to the interior of $\mathcal{P}_{\mathcal{I}_f}$. It means that an intersection point $\mathbf{v}$ which is not a vertex of $\mathcal{P}_{\mathcal{I}_f}$ will be visited which is impossible according to *Proof 1* above. Necessarily, we have $\mathcal{A}^{-1}(\Lambda) = \emptyset$. $\square$

E. Degenerate Cases

Degeneracy may happen when inequality lines corresponding to different rows of (5) are parallel or when three or more of these lines have a common intersection point.

The first degenerate case corresponds to Case 1. of Section III-C ($\mathbf{n}_k \mathbf{n}_{i_\perp}^T = 0$) and is not difficult to handle. The second one, three or more inequality lines intersecting at a given point, corresponds to the particular case $\alpha_l = 0$ in (8). Equivalently, with the notations of Section III-C, there exists $k \in \{1,\ldots,m\}\backslash\{i\}$ such that $\alpha_k = 0$. It means that the three lines L_i, L_j and L_k (where L_k is $L_{k,min}$ or $L_{k,max}$) are crossing at $\mathbf{v}_{ij}$. However, together with L_j (which was followed to reach $\mathbf{v}_{ij}$), only one of L_i or L_k is supporting the current feasible polygon $\mathcal{P}_{\mathcal{I}_{ij}}$ along one of its edges. This supporting line is to be followed to reach the next vertex of the current feasible polygon. It can be determined by straightforward geometric reasoning on the vectors $\mathbf{n}_i$, $\mathbf{n}_j$ and $\mathbf{n}_k$ [40]. The other line intersecting at $\mathbf{v}_{ij}$ supports the polygon at a vertex only. Removing its index from consideration in (8) yields a strictly positive value of α_l, i.e., it leads to the next polygon vertex.

F. Example

The determination of the vertices of $\mathcal{A}^{-1}(\Lambda)$ in the case of Fig. 3 is taken as an example. As shown in Fig. 5, the problem is feasible and the vertices $\mathbf{v}_7$, $\mathbf{v}_8$, $\mathbf{v}_9$, $\mathbf{v}_{10}$, and $\mathbf{v}_{11}$ of $\mathcal{A}^{-1}(\Lambda)$ are determined in a clockwise order. The initial intersection point $\mathbf{v}_{\text{init}}$ has been selected as the intersection between the inequality lines $L_{3,min}$ and $L_{6,max}$. This initial point is not feasible since its feasible index set is $\mathcal{I}_{init} = \{1,2,3,5,6\}$. The feasible index set at $\mathbf{v}_2$ is $\mathcal{I}_2 = \{1,2,3,4,5,6\}$ whereas the feasible index set at $\mathbf{v}_3$, $\mathbf{v}_4$, $\mathbf{v}_5$, and $\mathbf{v}_6$ is $\mathcal{I}_3 = \{1,2,3,4,5,6,8\}$. By construction of the algorithm, we have $\mathcal{I}_{init} \subset \mathcal{I}_2 \subset \mathcal{I}_3 \subset \{1,\ldots,8\}$.

IV. Optimal Tension Distributions

This section shows that various cable tension distributions can be obtained by a straightforward use or modification of the algorithm introduced in Section III. The determinations of the optimal 2-norm and 1-norm as well as the centroid and weighted barycenter tension distributions are explained. In the sequel, the feasible polygon $\mathcal{A}^{-1}(\Lambda)$ is assumed to be non-empty since, otherwise, no feasible tension distribution exists.

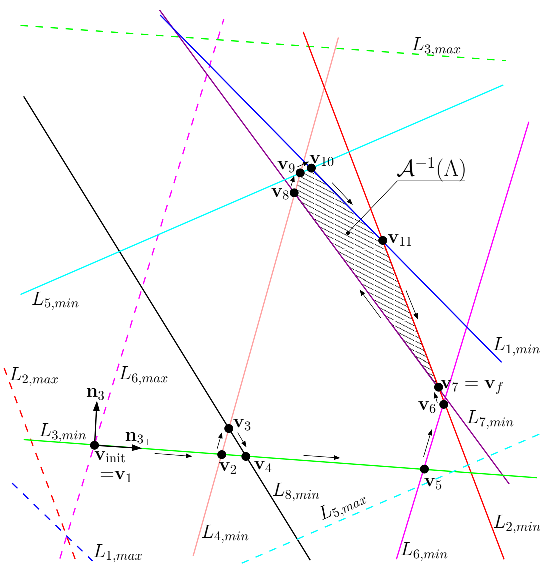


Fig. 5. Algorithm of Section III applied to the case of Fig. 3. From the initial intersection point $\mathbf{v}_{\text{init}}$, as indicated by the arrows, the algorithm reaches in order the intersection points $\mathbf{v}_2, \ldots, \mathbf{v}_{11}$. It finally terminates at $\mathbf{v}_7 = \mathbf{v}_f$ since this point was already visited before.

A. 2-norm Optimal Solution

The 2-norm optimal tension distribution is the solution of the quadratic program (QP)

$$\begin{aligned} &\min_{\mathbf{t}} \ \|\mathbf{t}\|^2 \\ &s.t. \quad \mathbf{W}\mathbf{t} = \mathbf{f}, \quad \mathbf{t}_{\min} \le \mathbf{t} \le \mathbf{t}_{\max}. \end{aligned} \tag{9}$$

Using Eq. (2), the objective function can be written as $\|\mathbf{t}\|^2 = \|\mathbf{t}_p\|^2 + \|\mathbf{N}\boldsymbol{\lambda}\|^2 + 2\mathbf{t}_p^T\mathbf{N}\boldsymbol{\lambda}$, i.e., $\|\mathbf{t}\|^2 = \|\mathbf{t}_p\|^2 + \|\boldsymbol{\lambda}\|^2$ since $\mathbf{N}^T\mathbf{N} = \mathbb{I}_2$ and $\mathbf{t}_p^T\mathbf{N} = \mathbf{0}$.

For a given pose of the mobile platform, $\mathbf{t}_p$ is a constant. The problem can thus be reformulated in the following equivalent form

$$\begin{aligned} &\min_{\boldsymbol{\lambda}} \ \|\boldsymbol{\lambda}\|^2 \\ &s.t. \quad \mathbf{t}_{\min} - \mathbf{t}_p \le \mathbf{N}\boldsymbol{\lambda} \le \mathbf{t}_{\max} - \mathbf{t}_p. \end{aligned} \tag{10}$$

This strictly convex QP has a unique global solution. When $\mathbf{0} \in \mathcal{A}^{-1}(\Lambda)$, $\boldsymbol{\lambda} = \mathbf{0}$ is the solution of (10) and $\mathbf{t}_p = \mathbf{W}^+\mathbf{f}$ is then the optimal solution of (9). When $\mathbf{0} \notin \mathcal{A}^{-1}(\Lambda)$, the optimal solution of (10) lies on an edge or at a vertex of $\mathcal{A}^{-1}(\Lambda)$ as illustrated in Fig. 6. The following straightforward modification of the algorithm introduced in Section III allows the determination of this solution. While turning around $\mathcal{A}^{-1}(\Lambda)$, the current vertex $\mathbf{v}_{ij}$ and the edge that will be followed to reach the next vertex $\mathbf{v}_{li}$ are tested for optimality. If this vertex or a point on this edge is found to be the optimal solution $\boldsymbol{\lambda}_2^*$ of (10), the desired 2-norm optimal tension distribution (solution of (9)) is computed as $\mathbf{t}_2^* = \mathbf{t}_p + \mathbf{N}\boldsymbol{\lambda}_2^*$.

By means of the Karush-Kuhn-Tucker (KKT) conditions [41], a vertex $\mathbf{v}_{ij}$ of $\mathcal{A}^{-1}(\Lambda)$ can be proved to be the solution of (10) if and only if the two components of the vector

$$\boldsymbol{\mu} = [\mathbf{a}_i, \mathbf{a}_j]^{-1}\mathbf{v}_{ij} \tag{11}$$

are non-negative. Vertex $\mathbf{v}_{ij}$ is the intersection between the two inequality lines L_i and L_j whose equations are $\mathbf{n}_i\boldsymbol{\lambda} = b_i - t_{p_i}$

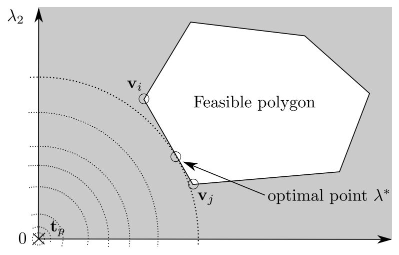
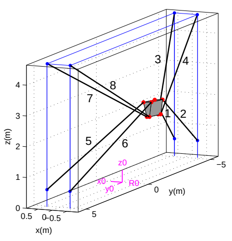
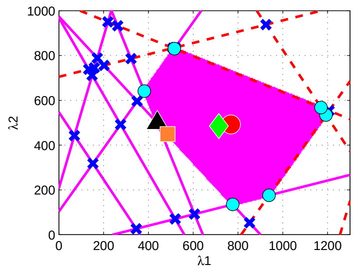


Fig. 6. A case where the solution of (10) lies on an edge of $\mathcal{A}^{-1}(\Lambda)$.



(a) Schematic view



(b) Feasible polygon $\mathcal{A}^{-1}(\Lambda)$ and various optimal points in the plane (λ_1, λ_2)

Fig. 7. The CABLAR fully constrained CDPR in static equilibrium at pose $[0\ -2.5\ \ 2\ \ 3\ \ 5\ \ 0]^T$ (units: meters and degrees, XYZ Euler angle convention) with a 45 kg platform mass, $\mathbf{t}_{\text{min}} = 100$ N and $\mathbf{t}_{\text{max}} = 600$ N. The markers indicate polygon vertices and different optimal solutions to the tension distribution problem.

and $\mathbf{n}_j\boldsymbol{\lambda} = b_j - t_{p_j}$, respectively. In (11), $\mathbf{a}_i = s_i\mathbf{n}_i^T$ where $s_i = 1$ if $b_i = t_{min}$ and $s_i = -1$ if $b_i = t_{max}$, and $\mathbf{a}_j = s_j\mathbf{n}_j^T$ where $s_j = 1$ if $b_j = t_{min}$ and $s_j = -1$ if $b_j = t_{max}$.

Let us now consider an edge of $\mathcal{A}^{-1}(\Lambda)$ supported by the inequality line $\mathbf{n}_i\boldsymbol{\lambda} = b_i - t_{p_i}$. The KKT conditions can be used to prove that the solution of (10) lies on this edge if and only if: 1. $\mu_i = s_i(b_i - t_{p_i})/\mathbf{a}_i^T\mathbf{a}_i \geq 0$, and 2. $\boldsymbol{\lambda} = \mu_i\mathbf{a}_i$ verifies all the inequalities in (5), where s_i and $\mathbf{a}_i$ are defined as above. If these two conditions are true, the point $\boldsymbol{\lambda} = \mu_i\mathbf{a}_i$ lies on the edge of $\mathcal{A}^{-1}(\Lambda)$ and is the optimal solution of (10): $\boldsymbol{\lambda}_2^* = \mu_i\mathbf{a}_i$. Geometrically, this point is the orthogonal projection of the origin $\boldsymbol{\lambda} = \mathbf{0}$ onto the line supporting the edge.

Fig. 7 illustrates different optimal solutions obtained for the CABLAR fully constrained CDPR. The square marker represents $\boldsymbol{\lambda}_2^*$ which lies on a edge of $\mathcal{A}^{-1}(\Lambda)$.

B. 1-norm Optimal Solution

The 1-norm optimal tension distribution is the solution of the linear program

$$\begin{aligned} \min_{\mathbf{t}} \ & \mathbf{1}^T\mathbf{t} \\ \text{s.t.} \quad & \mathbf{W}\mathbf{t} = \mathbf{f}, \quad \mathbf{t}_{\text{min}} \leq \mathbf{t} \leq \mathbf{t}_{\text{max}} \end{aligned} \tag{12}$$

where $\mathbf{1}$ is the vector of $\mathbb{R}^m$ whose components are all equal to 1. Eq. (2) allows (12) to be written in the equivalent form

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \ & \mathbf{1}^T\mathbf{N}\boldsymbol{\lambda} \\ \text{s.t.} \quad & \mathbf{t}_{\text{min}} - \mathbf{t}_p \leq \mathbf{N}\boldsymbol{\lambda} \leq \mathbf{t}_{\text{max}} - \mathbf{t}_p. \end{aligned} \tag{13}$$

since $\mathbf{1}^T\mathbf{t} = \mathbf{1}^T\mathbf{t}_p + \mathbf{1}^T\mathbf{N}\boldsymbol{\lambda}$. The solution of (13) lies at a vertex of $\mathcal{A}^{-1}(\Lambda)$. However, some orientations of $\mathcal{A}^{-1}(\Lambda)$ make the 1-norm minimized over an entire edge [41] and the solution is then non-unique. The use of the 1-norm optimal tension distribution is thus an issue because it may be discontinuous along a platform trajectory [26], [28], which might yield vibrations and lead to control issues.

Nevertheless, if the solution of (13) is to be computed, the algorithm introduced in Section III can be directly used since it suffices to test each visited vertex of $\mathcal{A}^{-1}(\Lambda)$ for optimality. Let $\mathbf{v}_{ij}$ be a vertex of $\mathcal{A}^{-1}(\Lambda)$ lying at the intersection between L_i and L_j whose equations are $\mathbf{n}_i\boldsymbol{\lambda} = b_i - t_{p_i}$ and $\mathbf{n}_j\boldsymbol{\lambda} = b_j - t_{p_j}$, respectively. The KKT conditions imply that $\mathbf{v}_{ij}$ is the optimal solution of (13) if and only if the vector

$$\boldsymbol{\mu} = [\mathbf{a}_i,\ \mathbf{a}_j]^{-1}\mathbf{N}^T\mathbf{1} \tag{14}$$

has non-negative components. In (14), $\mathbf{a}_i$ and $\mathbf{a}_j$ are defined as in Section IV-A. If $\mathbf{v}_{ij}$ is found be the solution of (13), the 1-norm optimal tension distribution (solution of (12)) is calculated as $\mathbf{t}_1^* = \mathbf{t}_p + \mathbf{N}\mathbf{v}_{ij}$. In Fig. 7b, the 1-norm minimum tension distribution is indicated by the triangular marker which, as expected, is located at a vertex of $\mathcal{A}^{-1}(\Lambda)$.

C. Centroid

The determination of the centroid $\boldsymbol{\lambda}_c^*$ of $\mathcal{A}^{-1}(\Lambda)$ can be relevant for suspended CDPRs since the corresponding tension distribution is far from the boundaries of Λ. In fact, according to [42], [43], the centroid is as far as possible from the polygon boundaries and is the unique point of $\mathcal{A}^{-1}(\Lambda)$ that solves

$$\max_{\mathbf{p}} \log \prod d_i \tag{15}$$

where d_i is the Euclidean distance from point $\mathbf{p}$ to the ith inequality line. Eq. (15) shows that the centroid depends smoothly on the problem constraints so that the corresponding tension distribution is continuous along a platform trajectory.

Let $\mathbf{v}_i = [v_{i_1}\ \ v_{i_2}]^T$, $i \in \{1, \ldots, q\}$, be the q vertices of $\mathcal{A}^{-1}(\Lambda)$ computed by means of the algorithm of Section III. Since these vertices have been determined in a clockwise or counter-clockwise order, the centroid $\boldsymbol{\lambda}_c^* = [\lambda_{c_1}\ \ \lambda_{c_2}]^T$ of $\mathcal{A}^{-1}(\Lambda)$ is directly given by the following well-known formulas

$$\begin{cases} \lambda_{c_1} = \frac{1}{6A}\sum\limits_{i=1}^{q-1}(v_{i_1} + v_{(i+1)_1})(v_{i_1}v_{(i+1)_2} - v_{(i+1)_1}v_{i_2}) \\ \lambda_{c_2} = \frac{1}{6A}\sum\limits_{i=1}^{q-1}(v_{i_2} + v_{(i+1)_2})(v_{i_1}v_{(i+1)_2} - v_{(i+1)_1}v_{i_2}) \end{cases} \tag{16}$$

where A is the area of the polygon, given by

$$A = \frac{1}{2}\sum_{i=1}^{q-1}(v_{i_1}v_{(i+1)_2} - v_{(i+1)_1}v_{i_2}). \tag{17}$$

In Fig. 7b, the red dot marker shows $\boldsymbol{\lambda}_c^*$. The desired tension distribution is the centroid of Λ given by $\mathbf{t}_c^* = \mathbf{t}_p + \mathbf{N}\boldsymbol{\lambda}_c^*$.

D. Weighted Barycenter

All the vertices of $\mathcal{A}^{-1}(\Lambda)$ being determined by the algorithm of Section III, the weighted barycenter is another tension distribution that can be directly computed. In order to avoid discontinuities that may be created when the same weight value is used for all vertices [44], each vertex $\mathbf{v}_i$ can be weighted by

$$w_i = \left(\sum_{j=1}^{2} \|\mathbf{v}_i - \mathbf{v}_{ij}\| \right) / \|\mathbf{v}_i\| \tag{18}$$

where $\mathbf{v}_{i1}$ and $\mathbf{v}_{i2}$ are the two neighbor vertices of $\mathbf{v}_i$. The weighted barycenter of $\mathcal{A}^{-1}(\Lambda)$ is then computed as

$$\boldsymbol{\lambda}_w = \left(\sum_{i=1}^{q} w_i \mathbf{v}_i \right) \Big/ \sum_{i=1}^{q} w_i \tag{19}$$

and the desired tension distribution is $\mathbf{t}_w = \mathbf{t}_p + \mathbf{N}\boldsymbol{\lambda}_w$. This solution is shown in Fig. 7b by the diamond marker.

V. Computational Efficiency

In this section, the maximum number of moves (iterations) made by the algorithm of Section III is determined. Based on this result, a detailed analysis of the number of operations required by the proposed tension distribution algorithm is presented together with brief comparisons to some previously proposed methods.

A. Number of Feasible Polygon Vertices

By geometric reasoning, let us prove that the number of vertices n_v of a feasible polygon $\mathcal{P}_\mathcal{I}$ satisfies $n_v \leq 2|\mathcal{I}|$, where $|\mathcal{I}|$ is the number of elements of $\mathcal{I}$. This result will be used in Section V-B.

In the case $|\mathcal{I}| = 2$, two pairs of parallel inequality lines are intersecting as shown in Fig. 8a. Without loss of generality, let the first pair be $L_{1,min}$, $L_{1,max}$ (row 1 of (5)) and the second pair be $L_{2,min}$, $L_{2,max}$ (row 2 of (5)). The resulting feasible polygon $\mathcal{P}_{\mathcal{I}=2}$ is a parallelogram so that $n_v = 4$ when $|\mathcal{I}| = 2$. In the case $|\mathcal{I}| = 3$, a third pair of parallel inequality lines $L_{3,min}$ and $L_{3,max}$ (row 3 of (5)) is added as shown in Fig. 8b. The corresponding feasible polygon $\mathcal{P}_{\mathcal{I}=3}$ has the maximum possible number of vertices when both $L_{3,min}$ and $L_{3,max}$ cut one and only one vertex out of $\mathcal{P}_{\mathcal{I}=2}$. Indeed, as illustrated in Fig. 8b, an additional inequality line creates new vertices by cutting some vertices out of the current polygon. The polygon being convex, at most two vertices can be created and at least one vertex is cut out in the process, i.e., at most one vertex is added. $L_{3,min}$ and $L_{3,max}$ add thus at most 2 vertices to the previous 4-vertex polygon $\mathcal{P}_{\mathcal{I}=2}$ so that $n_v \leq 6$ holds for $|\mathcal{I}| = 3$. By induction on the number of pairs of inequality lines defining $\mathcal{P}_\mathcal{I}$, the same reasoning leads to $n_v \leq 2|\mathcal{I}|$.

B. Maximum Total Number of Moves

Let $\mathbf{v}_{\text{init}}$ be the intersection point where the algorithm introduced in Section III starts. The feasible index set at $\mathbf{v}_{\text{init}}$ is denoted $\mathcal{I}_{\text{init}}$. On its way to the last intersection point $\mathbf{v}_f$, the algorithm considers *a sequence of feasible index sets* $\mathcal{I}(i)$.

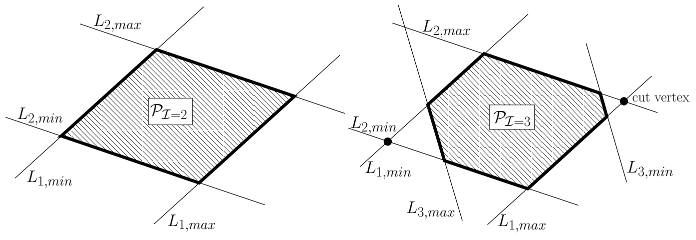


Fig. 8. Feasible polygons $\mathcal{P}_\mathcal{I}$ with $|\mathcal{I}| = 2$ (left) and $|\mathcal{I}| = 3$ (right).

Without loss of generality (reordering if necessary), let the first index set be $\mathcal{I}_{\text{init}} = \mathcal{I}(1) = \{1, \ldots, p\}$ where p is the number of rows of (5) satisfied at $\mathbf{v}_{\text{init}}$ ($p \leq m$). If (5) is feasible, the (last) feasible index set at $\mathbf{v}_f$ is $\mathcal{I}_f = \{1, \ldots, m\}$, otherwise it is $\mathcal{I}_f \subset \{1, \ldots, m\}$. The maximum number of feasible index sets in the sequence is thus obtained when (5) is feasible in which case $\mathcal{I}_f = \mathcal{I}(m-p+1) = \{1, \ldots, m\}$. The ith index set in the sequence is $\mathcal{I}(i) = \{1, \ldots, p, \ldots, p+i-1\}$, $1 \leq i \leq m-p+1$. The number of vertices of the corresponding feasible polygon $\mathcal{P}_{\mathcal{I}(i)}$ is denoted $n_v(i)$. When $\mathcal{I}(i)$ is the current feasible index set, the algorithm "moves" along the edges of $\mathcal{P}_{\mathcal{I}(i)}$ from one vertex to the next one. Let $n_m(i)$ be defined as the number of moves along the edges of $\mathcal{P}_{\mathcal{I}(i)}$ such that $\mathcal{I}(i)$ is the feasible index set of the vertices from where the moves are made ($1 \leq n_m(i) \leq n_v(i)$).

The maximum total number of moves (iterations) made by the algorithm of Section III is equal to

$$n_{mt} = \sum_{i=1}^{m-p+1} n_m(i). \tag{20}$$

An upper bound on n_{mt}, depending only on p and m, can be established by analyzing the relationship between the number of vertices $n_v(i+1)$ of $\mathcal{P}_{\mathcal{I}(i+1)}$, the number of vertices $n_v(i)$ of $\mathcal{P}_{\mathcal{I}(i)}$, and the number of moves $n_m(i)$ along the edges of $\mathcal{P}_{\mathcal{I}(i)}$. First, since $\mathcal{I}(i+1) = \mathcal{I}(i) \cup \{p+i\}$, $\mathcal{P}_{\mathcal{I}(i+1)}$ is obtained from $\mathcal{P}_{\mathcal{I}(i)}$ by cutting it with the two inequality lines $L_{p+i,min}$ and $L_{p+i,max}$ which correspond to the row $p+i$ of (5), as illustrated in the example shown in Fig. 9. Second, by definition of $n_m(i)$, the algorithm of Section III stops at $n_m(i)$ vertices of $\mathcal{P}_{\mathcal{I}(i)}$ before meeting one of the two inequality lines $L_{p+i,min}$ or $L_{p+i,max}$, i.e., before reaching a vertex of $\mathcal{P}_{\mathcal{I}(i+1)}$. Therefore, the inequality line $L_{p+i,min}$ or $L_{p+i,max}$ on which this vertex lies ($L_{p+i,max}$ in Fig. 9) cuts at least $n_m(i)$ vertices of $\mathcal{P}_{\mathcal{I}(i)}$ while creating two new vertices. The other inequality line ($L_{p+i,min}$ in Fig. 9) cuts at least one vertex of $\mathcal{P}_{\mathcal{I}(i)}$ while creating two new vertices. Hence, the number of vertices of $\mathcal{P}_{\mathcal{I}(i)}$ which are also vertices of $\mathcal{P}_{\mathcal{I}(i+1)}$ is at most equal to $n_v(i) - n_m(i) - 1$ and the number of vertices created by $L_{p+i,min}$ and $L_{p+i,max}$ is at most equal to four. The number of vertices $n_v(i+1)$ of $\mathcal{P}_{\mathcal{I}(i+1)}$ is thus bounded as follows

$$n_v(i+1) \leq n_v(i) - n_m(i) + 3\,. \tag{21}$$

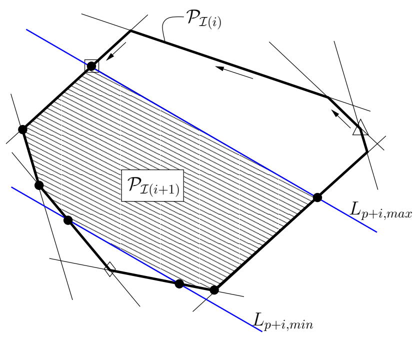

$\mathcal{P}_{\mathcal{I}(i)}$
$\mathcal{P}_{\mathcal{I}(i+1)}$
$L_{p+i,max}$
$L_{p+i,min}$

Fig. 9. The edges of $\mathcal{P}_{\mathcal{I}(i)}$ are shown in thick black lines. The feasible polygon $\mathcal{P}_{\mathcal{I}(i+1)}$ differs from $\mathcal{P}_{\mathcal{I}(i)}$ in that the two inequality lines shown in blue, $L_{p+i,min}$ and $L_{p+i,max}$, are bounding it. The vertices of $\mathcal{P}_{\mathcal{I}(i+1)}$ are indicated by black dots. The vertex marked with the triangle is the one from where the algorithm begins to follow the edges of $\mathcal{P}_{\mathcal{I}(i)}$. In this example, the algorithm makes three moves (indicated by the arrows) before reaching a vertex of $\mathcal{P}_{\mathcal{I}(i+1)}$. The latter vertex lies on $L_{p+i,max}$ and is shown by the square. $L_{p+i,max}$ creates two vertices and cuts four vertices of $\mathcal{P}_{\mathcal{I}(i)}$ whereas $L_{p+i,min}$ creates two vertices and cuts one vertex of $\mathcal{P}_{\mathcal{I}(i)}$ (the one shown by the diamond).

By induction on i, (21) leads directly to

$$n_v(i+1) \leq n_v(1) + 3i - \sum_{j=1}^{i} n_m(j) \tag{22}$$

and, since $n_m(i+1) \leq n_v(i+1)$, (22) implies that

$$\sum_{j=1}^{i+1} n_m(j) \leq n_v(1) + 3i\,. \tag{23}$$

According to Section V-A, $n_v(1) \leq 2p$ (since $|\mathcal{I}(1)| = p$) so that, with $i = m - p$ in (23), the following upper bound on the total number of moves is obtained

$$n_{mt} \leq 2p + 3(m-p) = 3m - p \tag{24}$$

where, in this paper, $m = n + 2$. When the number of DOFs is $n = 6$, the number of cables is $m = 8$ and $n_{mt} \leq 24 - p$ where p is the number of rows of (5) satisfied at $\mathbf{v}_{\text{init}}$ ($p \geq 2$), i.e., $n_{mt} \leq 22$.

When $p = m$, $\mathbf{v}_{\text{init}}$ is a vertex of $\mathcal{A}^{-1}(\Lambda)$ and $n_{mt} \leq 2m$, which corresponds to the case where the algorithm of Section III starts at a vertex of $\mathcal{A}^{-1}(\Lambda)$ and then makes a full turn around it. This case is consistent with Section V-A where the number of vertices of $\mathcal{A}^{-1}(\Lambda)$ is proved to be less than or equal to $2|\mathcal{I}_f| = 2m$. When $p \neq m$, the intersection point $\mathbf{v}_{\text{init}}$ where the algorithm starts is unfeasible and the upper bound established in (24) shows that, in the worst case, the total number of moves remains fairly small.

C. Maximum Number of Operations

The maximum number of moves of the algorithm introduced in Section III being determined, the maximum number of floating point operations (FLOPs) can be established. Here, a FLOP is an addition, subtraction, multiplication, division or square root.

The proposed tension distribution algorithm consists of the following main steps.

1. Initialization: Computations of $\mathbf{t}_p = \mathbf{W}^{+}\mathbf{f}$, of the nullspace matrix $\mathbf{N}$, and of the first intersection point $\mathbf{v}_{\text{init}}$.
2. Algorithm of Section III.
3. Computation of the desired tension distribution: One of the solutions presented in Section IV.

In this paper, the computation of $\mathbf{t}_p$ and $\mathbf{N}$ in step 1 is based on the QR decomposition of $\mathbf{W}^T$ obtained by means of Householder triangularizations [45]. Including the computation of $\mathbf{v}_{\text{init}}$ and being given that $m = n+2$, the corresponding number of FLOPs is equal to $\frac{8}{3}n^3 + \frac{33}{2}n^2 + \frac{167}{6}n$ which gives 1337 FLOPs for $n = 6$ DOF CDPRs.

A detailed analysis of Section III-C shows that the maximum number of FLOPs required for each move from one intersection point to the next one is equal to $10n + 17$. Hence, neglecting the small overhead occasionally required to handle degenerated cases (Section III-E), the number of FLOPs in step 2 is equal to $n_{mt}(10n + 17)$. According to Section V-B, the worst-case scenario is $n_{mt} = 22$ which leads to a maximum of 1694 FLOPs for $n = 6$. In practice, according to our experience on CaBLAR and CoGiRo and using a warm start (see Section V-D), the average maximum number of moves is $n_{mt} = 6$ which corresponds to 462 FLOPs.

In Section IV-C, the computation of the centroid tension distribution requires $10q + 4n$ FLOPs where q denotes the number of vertices of $\mathcal{A}^{-1}(\Lambda)$. In the worst case, $q = 2m = 2n+4$ so that the maximum number of FLOPs in Section IV-C is $24n + 40$ (184 for $n = 6$). In Section IV-D, the number of FLOPs involved in the computation of the weighted barycenter tension distribution is equal to $23q + 4n + 8$ FLOPs which, in the worst case $q = 2n + 4$, leads to $50n + 100$ FLOPs (400 for $n = 6$). In the case of the 2-norm and 1-norm optimal tension distributions, if (5) is found to be feasible, most of step 3 is actually done together with step 2 since optimality of a vertex of $\mathcal{A}^{-1}(\Lambda)$ (or a point on an edge in case of the 2-norm) is tested by means of the KKT conditions while turning around $\mathcal{A}^{-1}(\Lambda)$. However, the maximum number of operations is reached when this optimal point is located at the last vertex of $\mathcal{A}^{-1}(\Lambda)$ (or on its last edge in case of the 2-norm) visited by the algorithm of Section III. Therefore, the maximum number of operations is reached when the KKT conditions need to be tested at all the vertices of $\mathcal{A}^{-1}(\Lambda)$ and, in case of the 2-norm, for all its edges. Accordingly, the computation of the 1-norm optimal tension distribution in Section IV-B requires a maximum of $11q + 6n + 10$ which, in the worst case $q = 2n + 4$, leads to $28n + 54$ FLOPs (222 for $n = 6$). Moreover, the computation of the 2-norm optimal tension distribution in Section IV-A requires a maximum of $4nq + 22q + 4n + 8$ which, in the worst case $q = 2n+4$, leads to $8n^2 + 64n + 96$ FLOPs (768 for $n = 6$).

For $n = 6$ DOF CDPRs driven by $m = 8$ cables, *in the worst case* and according to the above analysis, the maximum total number of FLOPs required by the proposed tension distribution algorithm is equal to 3799, 3253, 3215, and 3431 in case of the 2-norm, 1-norm, centroid and weighted barycenter solutions, respectively. These numbers of FLOPs are strict upper bounds almost never attained in practice but they are very useful to check whether or not a real-time computation time constraint can be satisfied.

D. Warm Start

At a given pose along a discretized trajectory followed by the CDPR mobile platform, a possible warm start method consists in choosing $\mathbf{v}_{\text{init}}$ at the intersection point between the two inequality lines which intersected at $\mathbf{v}_f$ at the previous pose along the trajectory. It is worth noting that the cases in which this $\mathbf{v}_{\text{init}}$ is unfeasible are not an issue since the proposed algorithm can start at such an unfeasible point.

E. Comparison to Other Methods

As detailed at the end of Section II, the vertices of $\mathcal{A}^{-1}(\Lambda)$ can be simply computed by solving a number of 2×2 linear systems and testing the feasibility of their solutions [19], [35]. For $m = n + 2$, the corresponding number of FLOPs can be established to be $6n^3 + 52n^2 + 116n + 72$. This leads to 3936 FLOPs for $n = 6$ DOF CDPRs. Note that this number does not include the possible additional computations needed to order the vertices. The algorithm introduced in Section III-C involves 1694 FLOPs *in the worst case*. Hence, the simple determination of the vertices pointed out in [19], [35] requires *at least* more than twice as many FLOPs as the algorithm introduced in Section III. In practice, according to our experiments, it generally needs eight to twelve times as many FLOPs as the algorithm of Section III.

In [34], a method to compute the 2-norm optimal tension distribution is proposed. An upper bound (equal to 256 for $m = 8$) on the number of iterations is established when only the lower tension limit t_{min} is taken into account. When both t_{min} and t_{max} are considered, this upper bound on the number of iterations is given by $\sum_{s=0}^{2m} C_s^m$ which, for $m = 8$, is equal to 65536. At each iteration, a $m \times m$ or larger linear system must be solved so that, in the worst case, the number of required FLOPs is much larger than the worst case 3799 FLOPs needed by the algorithm proposed in the present paper to compute the 2-norm tension distribution.

Finally, the fast closed-form method proposed in [37] requires $-\frac{2}{3}n^3 + 2mn^2 - \frac{1}{2}n^2 + 8mn + 3m + \frac{25}{6}n$ FLOPs, i.e., 847 FLOPs for $m = n + 2 = 8$, when implemented with a QR decomposition based on Householder triangularizations. However, for this method to work in a large part of the WFW, in the case $m = n+2$, it may have to be called three times with one (resp. two) cable tension fixed to t_{min} or t_{max} during the second (resp. third) call [38]. Hence, the worst case number of FLOPs can be established to be $4n^3 + \frac{49}{2}n^2 + \frac{273}{6}n + 7$ FLOPs, i.e., 2026 FLOPs for $n = 6$ which is not significantly smaller than the worst-case number of FLOPs needed by the algorithm proposed in this paper.

VI. Experimental results

A. CDPR prototypes

1) CaBLAR *(Cable Based robot for Logistic Applications and Research):* This prototype, shown in Fig. 1, is a high-rack storage and retrieval machine which consists of a 6-DOF 8-cable fully constrained CDPR (2 DOR) with a mobile platform embedding a push-and-pull mechanism with a total mass of 45 kg. The push-and-pull mechanism allows the machine to automatically load and unload goods. The overall dimensions are $2\,\text{m} \times 12\,\text{m} \times 6\,\text{m}$ $(l \times L \times h)$. The real-time control system is based on Beckhoff TwinCAT3® and runs at a 1 kHz sampling frequency. The CPU of the real-time computer is a Intel® Core^{TM}2 Duo Processor T9400 @2.53 GHz.

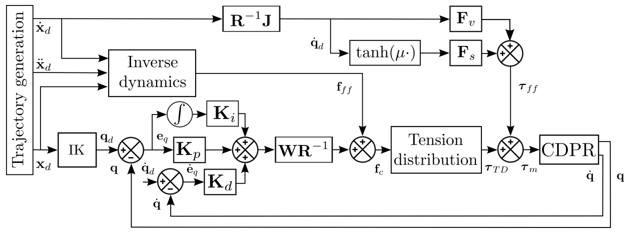


Fig. 10. Dual-space feedforward controller

2) CoGiRo*:* This prototype, shown in Fig. 2, is a 6-DOF 8-cable suspended CDPR (2 DOR) having a large workspace of overall dimensions $15\,\text{m} \times 11\,\text{m} \times 6\,\text{m}$ $(L \times l \times h)$. In the experiments reported in this paper, CoGiRo performs pick-and-place tasks, its mobile platform being equipped with a crane fork for a total mass of 93 kg. The control system is based on B&R Automation Studio® running at a 1 kHz sampling frequency with a CPU Intel® Core^{TM}2 Duo Processor L7400 @1.5 GHz.

The paper is accompanied by a video. It first shows the CaBLAR prototype moving along predefined trajectories. The CoGiRo prototype performing a pick-and-place task is then shown. Both CaBLAR and CoGiRo are controlled by means of the the dual-space feedforward controller of Section VI-B.

B. Dual-space feedforward controller

The dual-space feedforward control scheme shown in Fig. 10 is very similar to the one in [13]. The difference is the use of a joint space instead of an operational space PID controller, the tuning of the former being much simpler in practice.

In Fig. 10, as detailed in [13], the "Inverse dynamics" block compensates for the loaded mobile platform dynamics by means of an operational space feedforward wrench $\mathbf{f}_{ff}$ whereas the actuator dry and viscous frictions $\mathbf{F}_s$ and $\mathbf{F}_v$ are compensated for by a joint space feedforward torque $\boldsymbol{\tau}_{ff}$. The term $\tanh(\mu\dot{\mathbf{q}}_d)$ is used to model dry friction in order to avoid discontinuities which may appear with the sign function. Moreover, $\boldsymbol{\tau}_m$ is the actuator input torque vector and $\boldsymbol{\tau}_{TD}$ is the torque vector given by the tension distribution algorithm. Let us note that $\boldsymbol{\tau}_{TD} = \mathbf{R}\mathbf{t}_{TD}$, where $\mathbf{t}_{TD}$ is one of the tension distribution solutions of Section IV and $\mathbf{R}$ is a diagonal matrix whose nonzero components account for the mechanical transmission ratios and the winch drum radii. The input to the tension distribution block is $\mathbf{f}_c$ which is the sum of the feedforward wrench $\mathbf{f}_{ff}$ and a wrench generated by the PID controller [13].

The control scheme of Fig. 10 includes the tension distribution algorithm in the main control loop. If the latter algorithm finds a feasible tension distribution (i.e. (5) is feasible), the corresponding torque $\boldsymbol{\tau}_{TD}$ is computed. Hence, care must be taken to correctly tune the parameters used to calculate

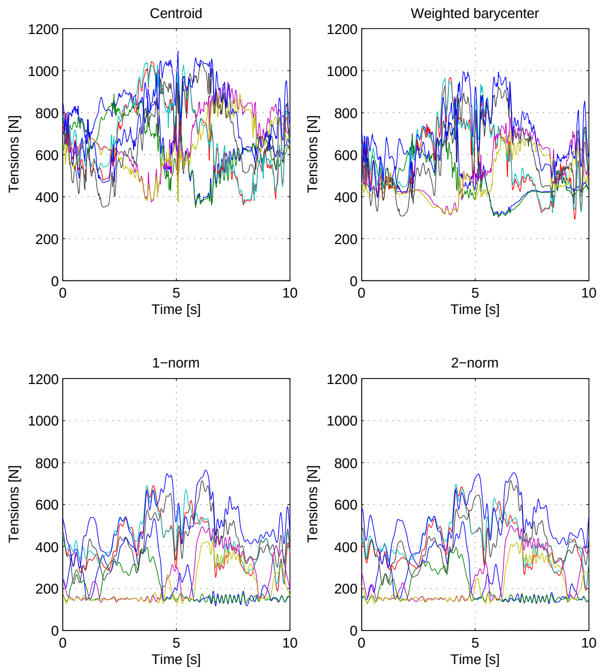


Fig. 11. Cable tensions measured on the CaBLAR prototype, obtained by means of four different tension distributions

the feedforward terms $\mathbf{f}_{ff}$ and $\boldsymbol{\tau}_{ff}$. Indeed, a bad tuning leads the PID controller to overwork making the input $\mathbf{f}_c$ to the tension distribution block impossible to balance with feasible cable tensions. Such a situation leads to a failure of the tension distribution algorithm (i.e. $\Lambda = \emptyset$), even if the current trajectory fully lies in the wrench-feasible workspace.

C. CaBLAR *real-time results*

The control scheme of Fig. 10 is applied to the fully constrained CDPR CaBLAR with $t_{\min} = 150\,\text{N}$ and $t_{\max} = 1200\,\text{N}$. The mobile platform motion is controlled along a path in the (y, z) plane (i.e. $x = 0$) with a constant null orientation. The y and z coordinates of the waypoints of the path are given in Tab. I. This path is representative of typical intralogistic applications.

TABLE I
CaBLAR: Waypoints of the desired trajectory

units: m	**0**	**1**	**2**	**3**	**4**
$[y\ z]$	$[0\ 2]$	$[-2.5, 1]$	$[-2.5, 2.5]$	$[2.5, 2.5]$	$[2.5, 1]$

The algorithm of Section III has been implemented and used to compute the four tension distributions presented in Section IV. Fig. 11 shows the measured cable tensions obtained with each method. The corresponding Root Mean Square (RMS) cable tension values are given in Tab. II. As expected, the 1-norm and 2-norm solutions result in lower cable tension values and their RMS tension values are very similar since, along the considered trajectory, $\boldsymbol{\lambda}_1^*$ and $\boldsymbol{\lambda}_2^*$ are always located at the same vertex or at nearby vertices of the polygon $\mathcal{A}^{-1}(\Lambda)$. The centroid results in the higher RMS cable tension. As shown in Fig. 11, the weighted barycenter provides an interesting alternative which gives a relatively low RMS tension value compared to the centroid while keeping the cable tensions relatively far from the boundaries of the feasible cable tension set Λ.

TABLE II
Root Mean Square cable tension values of the four considered tension distributions

	centroid	weighted barycenter	1-norm solution	2-norm solution
CaBLAR	700.9 N	569.6 N	356.5 N	354.7 N
CoGiRo	479.7 N	477.2 N	$\emptyset$	477.0 N

Tab. III shows the Task Execution Time (TET) of the four considered tension distributions, i.e., for each distribution, the total time needed for the computations of the three steps detailed in Section V-C. For comparison purposes, the TET of an efficient active set method (with warm start) that solves the QP (9) is also given. Tab. III illustrates that the proposed algorithm is efficient since the TETs are slightly larger but comparable to the TET of the active set method. Moreover, contrary to the latter which can only compute the 2-norm solution, the proposed algorithm is versatile since the four tension distributions in Tab. III were computed with the same computer code. To ensure real-time compatibility, the worst-case computation time should be determined. Let us assume that the TET is proportional to the number of FLOPs and let α denote the ratio of time divided by number of FLOPs. The centroid maximum TET given in Tab. III was obtained for $n_{mt} = q = 5$ which, according to Section V, corresponds to $1337 + 77n_{mt} + 10q + 24 = 1796$ FLOPs whereas the centroid average TET was in most cases obtained for $n_{mt} = q = 4$ which corresponds to 1709 FLOPs. The corresponding values of the ratio α are 0.0127 and 0.0119. The same analysis for the barycenter, 1-norm and 2-norm cases corroborate these values of α. In the remainder of this section, we thus choose $\alpha = 0.013$. Hence, the worst-case TET can be estimated from the maximum number of FLOPs given at the end of Section V-C by multiplying these number of FLOPs by $\alpha = 0.013$. The results are worst-case TETs of 49.4 µs, 42.3 µs, 41.8 µs, and 44.6 µs for the 2-norm, 1-norm, centroid and weighted barycenter solutions, respectively. This analysis proves that the proposed algorithm easily satisfies the real-time constraint which requires the TET to be smaller than 1 ms (1 kHz sampling frequency). No such worst-case bound is established for the active set algorithm whose TETs are shown in Tab. III. In case of the simple method [19], [35] analyzed at the beginning of Section V-E, an incompressible number of $1337 + 3936$ FLOPs, which corresponds to 68.5 µs for $\alpha = 0.013$, is required in order to compute $\mathbf{t}_p$, $\mathbf{N}$ and the vertices of $\mathcal{A}^{-1}(\Lambda)$. Adding the computations needed to calculate the desired tension distributions, the worst-case TET lies in the interval $72-80$ µs, i.e., four times as much as the TETs shown in Tab. III and twice as much as the worst-case TETs indicated above.

TABLE III
CaBLAR: Task Execution Time of the tension distributions

	average TET	min TET	max TET
centroid	20.3 µs	18.6 µs	22.8 µs
barycenter	21.3 µs	19.1 µs	24.3 µs
1-norm	17.7 µs	16.9 µs	19.1 µs
2-norm	17.7 µs	16.9 µs	18.9 µs
active set	14.1 µs	13.4 µs	15.8 µs

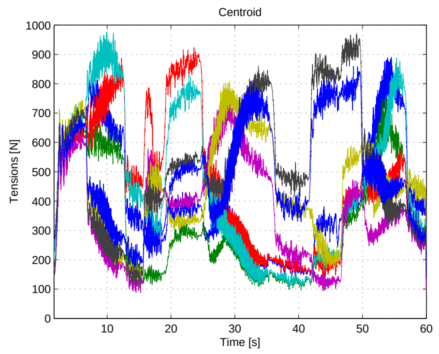


Fig. 12. Motor torques (converted in tensions) of the CoGiRo prototype obtained with the centroid tension distribution

D. CoGiRo *real-time results*

The CoGiRo suspended CDPR prototype is now considered with $t_{\text{min}} = 100\,\text{N}$ and $t_{\text{max}} = 5000\,\text{N}$. The dual-space feedforward controller of Fig. 10 is used to control the motion of the mobile platform along a trajectory described in Tab. IV, where $\mathbf{x} = [x\ y\ z\ \phi\ \theta\ \psi]^T$ (XYZ Euler angle convention) defines the pose of the mobile platform.

TABLE IV
CoGiRo: Waypoints of the mobile platform desired trajectory

$\mathbf{x}^T$ [m][°]	**0**	$[0\,0\,1\,0\,0\,0]$	**1**	$[-3.8\,1.2\,1\,0\,0\,-45]$
$\mathbf{x}^T$ [m][°]	**2**	$[-3.8\,1.2\,0.07\,0\,0\,-45]$	**3**	$[-3\,2\,0.07\,0\,0\,-45]$
$\mathbf{x}^T$ [m][°]	**4**	$[-3\,2\,1\,0\,0\,45]$	**5**	$[4\,-1\,0.5\,0\,0\,11]$
$\mathbf{x}^T$ [m][°]	**6**	$[4\,-1\,0.07\,0\,0\,11]$	**7**	$[4.3\,-2\,0.07\,0\,0\,11]$
$\mathbf{x}^T$ [m][°]	**8**	$[0\,0\,0.5\,0\,0\,0]$	**9**	$[0\,0\,0\,0\,0\,0]$

Nearby the center of the workspace of CoGiRo, an edge of the polygon $\mathcal{A}^{-1}(\Lambda)$ turns out to be almost aligned with the contours of the 1-norm objective function (Section IV-B). Consequently, the 1-norm tension distribution is not suitable because the optimal point switches very frequently from one vertex of $\mathcal{A}^{-1}(\Lambda)$ to another. This results in discontinuities in the tension distribution evolutions along the trajectory which generate significant platform and cable vibrations.

For the suspended CDPR CoGiRo, along the typical trajectory given in Tab. IV, the polygon $\mathcal{A}^{-1}(\Lambda)$ is small so that the three solutions $\boldsymbol{\lambda}_2^*$, $\boldsymbol{\lambda}_c^*$ and $\boldsymbol{\lambda}_w$ are almost identical along the trajectory. Hence, Fig. 12 only shows the motor torques—in terms of cable tensions—in case of the centroid tension distribution ($\boldsymbol{\lambda}_c^*$). The corresponding RMS cable tension values are given in Tab. II. From a general point of view, the centroid may be considered to be the best tension distribution for suspended CDPRs since $\boldsymbol{\lambda}_c^*$ is far from the boundaries of $\mathcal{A}^{-1}(\Lambda)$, i.e., slack cables should be avoided. However, in the present case of CoGiRo moving along the trajectory given in Tab. IV, selecting the pseudo-inverse solution $\mathbf{t}_p$ of Eq. (1) as the desired cable tension distribution is sufficient since the other tension distribution solutions considered in this paper leads to almost the same cable tensions as those in $\mathbf{t}_p$.

TABLE V
CoGiRo: Task Execution Time of the tension distributions

	average TET	min TET	max TET
centroid	24.9 µs	22.1 µs	28.3 µs
barycenter	25.8 µs	23.5 µs	29.5 µs
1-norm	21.1 µs	19.7 µs	20.2 µs
2-norm	18.2 µs	17.8 µs	18.3 µs
centroid [35]	74.9 µs	75.6 µs	76.8 µs

Tab. V shows the Task Execution Time (TET), i.e., for each of the four considered tension distributions, the total time needed for the computations of the three steps detailed in Section V-C. Let us assume again that the computation time is proportional to the number of FLOPs. The computation of $\mathbf{t}_p$ and $\mathbf{N}$ (step 1 in Section V-C) takes approximately $18\,\mu$s. Hence, the constant ratio α of time divided by number of FLOPs is considered to be equal to 0.0135. Accordingly, the worst-case TETs are $51.7\,\mu$s, $43.9\,\mu$s, $43.4\,\mu$s, and $46.3\,\mu$s for the 2-norm, 1-norm, centroid and weighted barycenter solutions, respectively. These computation time upper bounds satisfy the real-time constraint at $1\,$kHz. Finally, the last line of Tab. V shows the TET of the centroid tension distribution computation described in [35]. We did our best to implement it as efficiently as possible in the case $m = n + 2 = 8$, using notably a simple 2D triangulation method [46] (p. 18). These TETs are consistent with the computational cost analysis of this method given at the beginning of Section V-E.

VII. Conclusion

This paper introduced a tension distribution algorithm dedicated to n-DOF CDPRs redundantly actuated by $n+2$ cables. The proposed algorithm first computes the vertices of the convex polygon of feasible cable tension distributions by following the polygon edges in a clockwise or counterclockwise order, or it proves that this polygon is empty. Along the way or once all the polygon vertices are determined, it was then pointed out that the optimal 2-norm and 1-norm as well as the centroid and weighted barycenter tension distributions can be directly determined. This results in a versatile algorithm capable of determining various cable tension distributions. Moreover, the proposed algorithm can start at an unfeasible point which eases its use and, importantly, allowed its worst-case computational cost to be assessed. Indeed, the maximum total number of iterations of the algorithm was established and proved to be fairly small. Based on this result, a detailed analysis of the maximum number of floating point operations of the algorithm was provided and compared to some previous methods. This analysis proved the efficiency of the proposed algorithm. This efficiency was also verified by experimentations on the two 6-DOF 8-cable CDPR prototypes CaBLAR and CoGiRo.

The proposed algorithm is relevant for real-time implementations since it is at the same time versatile, self-contained, and efficient even in the worst-case. However, this algorithm is dedicated to the case of n-DOF CDPRs driven by $n + 2$ cables. While this class of CDPRs is highly relevant in many applications, it does not encompass all possible CDPR designs. Hence, it may be worth extending the proposed algorithm to $n + 3$ or more cables, even if this extension is not straightforward because the results obtained in this paper were mainly based on 2D geometric reasoning.

References

[1] J. Albus, R. Bostelman, and N. Dagalakis, "The NIST ROBOCRANE," *J. Robotic Syst.*, vol. 10, no. 5, pp. 709–724, 1993.

[2] W.-J. Shiang, D. Cannon, and J. Gorman, "Optimal Force Distribution Applied to a Robotic Crane with Flexible Cables," in *IEEE Int. Conf. Robot. Autom.*, San Francisco, CA, USA, 2000.

[3] J.-P. Merlet and D. Daney, "A portable, modular parallel wire crane for rescue operations," in *IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, USA, 2010, pp. 2834–2839.

[4] R. L. Williams II, "Cable-suspended haptic interface," *Int. J. of Virtual Reality*, vol. 3, no. 3, pp. 13–21, 1998.

[5] L. Dominjon, J. Perret, and A. Lécuyer, "Novel Devices and Interaction Techniques for Human-Scale Haptics," *Visual Comput.*, vol. 23, no. 4, pp. 257–266, Mar. 2007.

[6] D. Surdilovic and R. Bemhardt, "STRING-MAN: A New Wire Robot for Gait Rehabilitation," in *IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, USA, 2004.

[7] G. Rosati, P. Gallina, and S. Masiero, "Design, Implementation and Clinical Tests of a Wire-Based Robot for Neurorehabilitation," *IEEE Trans. Neural Sys. Rehab. Eng.*, vol. 15, pp. 560–569, 2007.

[8] Y. Mao and S. K. Agrawal, "Wearable Cable-driven Upper Arm Exoskeleton-Motion with Transmitted Joint Force and Moment Minimization," in *IEEE Int. Conf. Robot. Autom.*, Anchorage, AK, USA, 2010, pp. 4334–4339.

[9] C. Lambert, M. Nahon, and D. Chalmers, "Implementation of an Aerostat Positioning System With Cable Control," *IEEE/ASME Transactions on Mechatronics*, vol. 12(1), pp. 32–40, 2007.

[10] B. Y. Duan, Y. Y. Qiu, F. S. Zhang, and B. Zi, "On design and experiment of the feed cable-suspended structure for super antenna," *Mechatronics*, vol. 19, pp. 503–509, 2009.

[11] T. Bruckmann, W. Lalo, C. Sturm, D. Schramm, and M. Hiller, "Design and Realization of a High Rack Storage and Retrieval Machine based on Wire Robot Technology," in *Proc. Int. Symp. Dyn. Prob. Mech.*, Buzios, Rio de Janeiro, Brazil, 2013.

[12] M. Gouttefarde, J.-F. Collard, N. Riehl, and C. Baradat, "Geometry selection of a redundantly actuated cable-suspended parallel robot," *IEEE Trans. on Robotics*, vol. 31, no. 2, pp. 501–510, 2015.

[13] J. Lamaury and M. Gouttefarde, "Control of a Large Redundantly Actuated Cable-Suspended Parallel Robot," in *IEEE Int. Conf. on Robot. and Autom.*, Karlsruhe, Germany, 2013, pp. 4644–4649.

[14] S. Kawamura and K. Ito, "A New Type of Master Robot for Teleoperation Usign a Radial Wire Drive System," in *IEEE/RSJ Int. Conf. Intel. Robots Syst.*, Yokohama, Japan, 1993, pp. 26–30.

[15] M. Hiller, S. Fang, S. Mielczarek, R. Verhoeven, and D. Franitza, "Design, Analysis and Realization of Tendon-Based Parallel Manipulators," *Mech. and Mach. Theory*, vol. 40, pp. 429–445, 2005.

[16] S. Kawamura, W. Choe, S. Tanaka, and S. R. Pandian, "Development of an ultrahigh speed robot falcon using wire drive system," in *Proc. IEEE Int. Conf. Robotics and Automation*, Nagoya, Japan, 1995, pp. 215–220.

[17] S. Tadokoro, Y. Murao, M. Hiller, R. Murata, H. Kohkawa, and T. Matsushima, "A Motion Base With 6-DOF by Parallel Cable Drive Architecture," *IEEE/ASME Trans. Mech.*, vol. 7, no. 2, pp. 115–123, 2002.

[18] S. Fang, D. Franitza, M. Torlo, F. Bekes, and M. Hiller, "Motion control of a tendon-based parallel manipulator using optimal tension distribution," *IEEE/ASME Trans. on Mechatronics*, vol. 29, no. 3, pp. 561–568, Sep. 2004.

[19] S.-R. Oh and S. K. Agrawal, "Cable Suspended Planar Robots With Redundant Cables: Controllers with Positive Tensions," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 457–465, 2005.

[20] J.-P. Merlet, "Kinematics of the Wire-Driven Parallel Robot MARIONET Using Linear Actuators," in *IEEE Int. Conf. Robot. Autom.*, Pasadena, CA, USA, 2008, pp. 3857–3862.

[21] A. Pott, H. Mutherich, W. Kraus, V. Schmidt, P. Miermeister, and A. Verl, "IPAnema: A family of cable-driven parallel robots for industrial applications," in *Cable-Driven Parallel Robots*, T. Bruckmann and A. Pott, Eds. Springer, 2013, pp. 119–134.

[22] W. B. lim, S. H. Yeo, and G. Yang, "Optimization of tension distribution for cable-driven manipulators using tension-level index," *IEEE/ASME Trans. on Mechatronics*, vol. 19, no. 2, pp. 676–683, Apr. 2014.

[23] G. Abbasnejad and M. Carricato, "Direct geometrico-static problem of underconstrained cable-driven parallel robots with n cables," *IEEE Trans. on Robotics*, vol. 31, no. 2, pp. 468–478, 2015.

[24] P. Bosscher, A. T. Riechel, and I. Ebert-Uphoff, "Wrench-feasible workspace generation for cable-driven robots," *IEEE Trans. on Robotics*, vol. 22, no. 5, pp. 890–902, Oct. 2006.

[25] M. Gouttefarde, D. Daney, and J.-P. Merlet, "Interval-analysis-based determination of the wrench-feasible workspace of parallel cable-driven robots," *IEEE Trans. on Robotics*, vol. 27, no. 1, pp. 1–13, 2011.

[26] P. H. Borgstrom, B. L. Jordan, G. S. Sukhatme, M. A. Batalin, and W. J. Kaiser, "Rapid Computation of Optimally Safe Tension Distributions for Parallel Cable-Driven Robots," *IEEE Trans. on Robotics*, vol. 25, no. 6, pp. 1271–1281, Dec. 2009.

[27] C. Gosselin, "On the Determination of the Force Distribution in Overconstrained Cable-Driven Parallel Mechanisms," in *2nd Int. Work. Fund. Issues Future Res. Dir. Parallel Mech. Manip.*, N. Andreff, O. Company, M. Gouttefarde, O. Krut, and F. Pierrot, Eds., Montpellier, France, 2008, pp. 9–17.

[28] R. Verhoeven, "Analysis of the Workspace of Tendon-Based Stewart Platforms," Ph.D. dissertation, Universität Duisburg-Essen, 2004.

[29] R. Verhoeven and M. Hiller, "Tension distribution in tendon-based stewart platforms," in *Advances in Robot Kinematics*, J. Lenarčič and F. Thomas, Eds. Springer, 2002, pp. 117–124.

[30] T. Bruckmann, A. Pott, M. Hiller, and D. Franitza, "A Modular Controller for Redundantly Actuated Tendon-Based Stewart Platforms," in *EuCoMes, 1st Conf. Mech. Sc.*, Obergurgl, Austria, 2006.

[31] M. Agahi and L. Notash, "Redundancy Resolution of Wire-Actuated Parallel Manipulators," *Trans. Can. Soc. Mech. Eng.*, vol. 33, no. 4, pp. 561–573, 2009.

[32] M. J.-D. Otis, S. Perreault, T.-L. Nguyen-Dang, P. Lambert, M. Gouttefarde, D. Laurendeau, and C. M. Gosselin, "Determination and Management of Cable Interferences Between Two 6-DOF Foot Platforms in a Cable-Driven Locomotion Interface," *IEEE Trans. Sys., Man, Cyb. - Part A: Syst. Hum.*, vol. 39, no. 3, pp. 528–544, May 2009.

[33] M. Hassan and A. Khajepour, "Analysis of Bounded Cable Tensions in Cable-Actuated Parallel Manipulators," *IEEE Trans. on Robotics*, vol. 27, no. 5, pp. 891–900, 2011.

[34] H. D. Taghirad and Y. B. Bedoustani, "An Analytic-Iterative Redundancy Resolution Scheme for Cable-Driven Redundant Parallel Manipulators," *IEEE Trans. Robot.*, vol. 27, no. 6, pp. 1137–1143, Dec. 2011.

[35] L. Mikelsons, T. Bruckmann, M. Hiller, and D. Schramm, "A Real-Time Capable Force Calculation Algorithm for Redundant Tendon-Based Parallel Manipulators," in *IEEE Int. Conf. Robot. Autom.*, Pasadena, CA, USA, May 2008.

[36] P. Lafourcade, "Étude des manipulateurs parallèles à câbles, conception d'une suspension active pour soufflerie," Ph.D. dissertation, Thèse de Docteur Ingénieur de l'ENSAE, 2004.

[37] A. Pott, T. Bruckmann, and L. Mikelsons, "Closed-form Force Distribution for Parallel Wire Robots," in *Computational Kinematics*. Duisburg, Germany: Springer, 2009, pp. 25–34.

[38] A. Pott, "An Improved Force Distribution Algorithm for Over-Constrained Cable-Driven Parallel Robots," in *Computational Kinematics*, Barcelona, Spain, 2013, pp. 1–8.

[39] K. Muller, C. Reichert, and T. Bruckmann, "Analysis of a real-time capable cable force computation method," in *Cable-Driven Parallel Robots*, T. Bruckmann and A. Pott, Eds. Springer, 2015, pp. 227–238.

[40] J. Lamaury and M. Gouttefarde, "A tension distribution method with improved computational efficiency," in *Cable-Driven Parallel Robots*, T. Bruckmann and A. Pott, Eds. Springer, 2013, pp. 71–85.

[41] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.

[42] G. Sonnevend, "An "Analytical Centre" for Polyhedrons and New Classes of Global Algorithms for Linear (Smooth, Convex) Programming," in *Syst. Model. and Optim. Proc. 12th IFIP Conf.*, A. Prékopa, J. Szelezsáan, and B. Strazicky, Eds. Budapest, Hungary: Springer, 1985, pp. 866–875.

[43] M. J. Kaiser and T. L. Morin, "Characterizing Centers of Convex Bodies via Optimization," *Jour. Math. Anal. App.*, vol. 184, pp. 553–559, 1994.

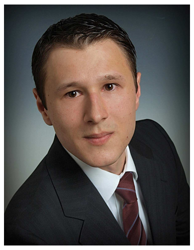
[44] T. Bruckmann, L. Mikelsons, T. Brandt, D. Schramm, A. Pott, and M. Abdel-Maksoud, "A novel tensed mechanism for simulation of maneuvers in wind tunnels," in *ASME Int. Design Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, 2009, pp. 17–24.

[45] L. M. Trefethen and D. Bau III, *Numerical Linear algebra*. SIAM, 1997.

[46] J. O'Rourke, *Computational Geometry in C*, 2nd ed. Cambridge University Press, 1998.

Marc Gouttefarde (S'03-M'06) received the B.Eng. degree in mechatronics from the Institut National des Sciences Appliquées (INSA), Strasbourg, France, in 2001, and the M.Sc. and Ph.D. degrees in mechanical engineering from Laval University, Québec, Canada, in 2002 and 2005, respectively.
From 2005 to 2007, he was a Postdoctoral Fellow at INRIA with the COPRIN project-team, Sophia-Antipolis, France. He is currently a CNRS Tenured Research Scientist with the LIRMM, Montpellier, France. His main research interests include the design and control of parallel manipulators and parallel cable-driven robots.

Johann Lamaury received the M.Eng. degree in mechatronics engineering from the Institut National des Sciences Appliquées (INSA), Strasbourg, France, in 2010 and the M.Sc. in control and robotics from the Télécom Physique Strasbourg school in 2010. He received the Ph.D. degree in robotics at the Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM), Montpellier, France, in 2013. He is currently working as a robotics engineer with the company Symétrie, Nîmes, France. His main research interests include the design and control of parallel manipulators.



Christopher Reichert received the B.Eng. and M.Eng. degrees in mechanical engineering from the University of Heilbronn, Germany, in 2010 and 2012, respectively. He is currently working toward the Ph.D. degree with special focus on cable-driven parallel robots at the Chair of Mechatronics, University of Duisburg-Essen, Germany. His research interests include impedance and force control, reconfiguration and energy-optimal trajectory planning.

Tobias Bruckmann received the Dipl.-Ing. in Mechanical Engineering from the University Duisburg-Essen, Duisburg, Germany, in 2004 and the Dr.-Ing. in 2010. He is currently working as a Senior Engineer at the Chair for Mechatronics, University Duisburg-Essen, where he is leading a research team with experiences in numerous fields of robotics, including cable-driven parallel manipulators, construction machines and human-machine interaction. His interests focus on cable-driven parallel manipulators, mechatronic system design and real-time control.