A Boundary Computation Algorithm for the Workspace Evaluation of Continuum Parallel Robots

Federico Zaccaria, Edoardo Idá, Sébastien Briot

A Boundary Computation Algorithm for the Workspace Evaluation of Continuum Parallel Robots

Federico Zaccaria *
University of Bologna,
Department of Industrial Engineering,
40131 Bologna, Italy
École Centrale de Nantes,
Laboratoire des Sciences du Numérique de Nantes
(LS2N), UMR CNRS 6004,
44321 Nantes, France
Email: federico.zaccaria3@unibo.it

Edoardo Idá
University of Bologna,
Department of Industrial Engineering,
40131 Bologna, Italy
Email: edoardo.ida2@unibo.it

Sébastien Briot
Centre National de la Recherche Scientifique (CNRS),
Laboratoire des Sciences du Numérique (LS2N) de Nantes
UMR CNRS 6004,
44321 Nantes, France
Email: Sebastien.Briot@ls2n.fr

*In this paper, a new algorithm for the computation of workspace boundaries of continuum parallel robots (*CPRs*) is proposed. State-of-the-art techniques are mainly based on time-consuming joint space discretization approaches or task-space discretization algorithms, and only a few approaches are dedicated to the computation of workspace boundaries. The proposed approach for the computation of the workspace boundaries, is based on i) a free-space exploration strategy and ii) a boundary reconstruction algorithm. The former is exploited to identify an initial workspace boundary location (exterior, interior boundaries, and holes), while the latter is used to reconstruct the complete boundary surface. Moreover, the algorithm is designed to be employed with* CPRs *modelling strategies based on general discretization assumptions, in order to increase its applicability for various scopes. Our method is compared with two state-of-the-art algorithms in four cases studies, to validate the results, and to establish its merits and limitations.*

1 Introduction

Continuum robots (*CRs*) are manipulators usually made of slender flexible components, developed to respond to the increasing necessity of safe interactions between robots and the environment. *CRs* are well suited for applications where the stiffness of rigid-link robots is considered a disadvantage, such as surgical tasks [1], and applications with mandatory safety requirements [2].

Continuum parallel robots (*CPRs*) have been proposed to mitigate the disadvantages of serial continuum robots (*SCRs*) [3], such as their reduced payload capability [4]. *CPRs* are commonly made by flexible beams disposed in a parallel arrangement, and connected to a rigid end-effector (*EE*). Possible applications of *CPRs* may include medium-to-large-scale applications where interaction and safety requirements are mandatory, and the robot environment is shared with humans: these applications may benefit from the flexibility-by-design and the reduced overall mass of *CPRs*.

Despite the great effort dedicated to *CRs* accurate modelling, research on their design is still complex. Several aspects need to be considered, such as the choice of actuation systems [5], beam arrangement [6], material selection [7], and performances [8]: the resulting architecture may considerably differ from the Gough-Stewart-like platforms firstly proposed in [3]. Workspace computation [9], namely the process of identifying a set of manipulator poses that satisfy given criteria, is a mandatory step toward the solution of *CPRs* design problems.

In *CPRs*, like any *CRs*, the geometry of the manipulator is not sufficient to describe the pose of the robot, and its configuration is also defined by the elastic deformation of its links. Thus, the equilibrium-configuration problem is said to be geometrico-static, and the static workspace evaluation of *CPRs* becomes complex since forward and inverse prob-

*Corresponding author.

lems do not admit an analytical solution, in general. *CPRs* workspace may be limited by several phenomena: *CPRs* may admit singular configurations that define the workspace limits but also stable-to-unstable transitions [10], [11]. Moreover, mechanical limits (e.g active of passive joint limits), material-strain limits, and mechanical interference may additionally reduce the robots workspace, and a workspace computation algorithm should consider all these aspects.

CRs workspace computation algorithms are generally based on two fundamental tools: the solution of a geometrico-static problem over different inputs (inverse or forward problems), and an algorithm dedicated to exploring workspace configuration candidates, which varies depending on the workspace computation needs. The choice of the exploration technique and the modelling strategy is usually the real bottleneck in the computational performance of the workspace evaluation tool. In general, *CRs* workspace computation algorithms should desirably be able to i) be generally applicable to different robot architectures and loading conditions, ii) find a suitable trade-off between accuracy and computational time, iii) consider mechanical and physical limits, and iv) assess singularity loci and equilibrium stability.

1.1 Brief state-of-the-art

In this Section, a brief state-of-the-art analysis of workspace computation algorithms of *CRs* is reported, to properly place our work over the current state-of-the-art, and to discuss the advantages and drawbacks of existing approaches. In this paper, we distinguish between algorithms that evaluate the full workspace of *CRs* (Section 1.1.1), and algorithms designed to find workspace boundaries (Section 1.1.2).

1.1.1 Full workspace computation algorithms

Full workspace computation algorithms evaluate the *CR* workspace by calculating each robot configuration satisfying workspace inclusion criteria. In the literature, two approaches can be mainly distinguished for exploring workspace candidates: i) actuation space sampling, and ii) task space sampling techniques.

The former is based on the discretization of the *CRs* actuation space and the iterative solution of a forward geometrico-static problem (*FGSP*). These algorithms bring simplicity and reduced complexity, while the main limit is related to their computational time, that can be significantly high, since it increases with the sampling density as well as the number of actuators [12]. Additionally, strain limits are considered only after the workspace computation, leading to unnecessary computations. Actuation-space sampling approaches have been used for the workspace computation of a pneumatically actuated *SCR* [13], concentric tube *SCRs* [14], and for a 6-degrees-of-freedom (*DoF*) *CPR* [15]. Similarly to geometrical approaches of rigid-link parallel robots, an approximation of the *CPRs* workspace was obtained by sampling the actuation space of each leg, computing its workspace, and obtaining the *CPR* workspace as the intersection of each leg workspace [16], [17]. However, the wrench exchanged between each arm and the *EE* cannot be considered and this may be a strong source of inaccuracy in the obtained results.

Task-space sampling techniques are based on the discretization of the *CRs* task space and the iterative solution of the inverse geometrico-static problem (*IGSP*). This way, the computational time can be reduced with respect to (w.r.t.) actuation-sampling strategies, since the overall time depends only on the sampling density but not on the number of actuators [12]. Preliminary results on the workspace evaluation of a 2-*DoF CPR* were shown in [18] with the results strictly limited to the planar case. A task-space sampling approach was employed in [19] for a 3-*DoF* planar *CPR*, and in [20] for a spatial *CPR* with an intermediate platform, but singularity analysis and equilibrium stability assessment are not performed. Similarly, singularity identification was preliminary performed in [21] during workspace computation thanks to a simplified mathematical model, but the accuracy may be reduced when the robot does not fit the constant curvature modelling assumptions. Flooding algorithms were employed in [10] for planar *CPRs* with a focus on singularity identification, and equilibrium stability assessment. An adaptive grid flooding algorithm was introduced in [22] to reduce the computational time of [10]. The certification of the numerical results was also discussed with the aim of preserving the same working mode [23] of the robot during the computation. Finally, a task space exploration strategy has recently been proposed in [24], with the workspace computation algorithm being based on the generation of several trajectories in the task space, and the iterative solution of the *IGSP* over these trajectories. Computationally efficiency is reached, but the identification of singularity loci has not been considered.

We may conclude that full workspace computation algorithms are applicable for general *CRs*. The computational time may be high, in particular if an actuation sampling strategy is employed. Even if most of the workspace exploration strategies are not dependent on the *CR* modelling strategy, the latter influences the capability to evaluate several features, such as singularities and equilibrium stability.

1.1.2 Boundary workspace computation algorithms

Boundary workspace computation algorithms are usually employed when full workspace computation algorithms require significant computational time. In contrast to full workspace computation approaches, workspace boundary computation algorithms aim to reconstruct the external and internal borders of the robot workspace only. They are usually faster than full workspace computation approaches, since they do not need to completely explore the task space. In the case of *CRs*, boundary-computation strategies are mainly based on i) the identification of closed-form analytical solutions, ii) continuation approaches, and iii) optimization approaches.

The closed-form equations of the workspace boundary may often be identified for rigid-link robots [25]. However, the complexity of the *CR* models makes such an identifica-

tion particularly challenging, and exact solutions were found only in a few cases [26].

Continuation algorithms [27] provide efficient tools, but they are generally limited to planar cases and three-dimensional workspaces are obtained only by superimposition of several planar slices. Additionally, inequality constraints (e.g. related to stability assessment or material limits) are hard to include, since they usually require to reformulate the problem as a set of equivalent equalities with the use of slack variables. Multiple boundaries of the workspace (such as voids or holes) are tricky to identify as well: the continuation approaches require the identification of a first point of the workspace boundary, and several boundaries are usually identified by an *ad-hoc* selection of this initial guess. A continuation approach is used in [28] for the boundary workspace computation of tendon-driven *SCRs*.

Finally, optimization approaches are a class of tools for the computation of workspace boundaries, firstly introduced in [29] for rigid-link robots. These approaches find boundary location by solving a constrained optimization problem, where the cost function is the distance between the robot end-effector and a user-defined point placed out of the workspace, while the constraints include the robot equilibrium equations. Optimization approaches lead to quite general algorithms, and several equality and inequality conditions can be additionally included in the constraints set. However, in case the robot admits multiple working modes [23], the numerical optimization may be directed toward a different working mode that provides a lower value of the cost function, which leads to incorrect solutions as it will be illustrated further. As for continuation approaches, voids are difficult to detect. Optimization algorithms for *CRs* were employed in [30], [31].

In conclusion, boundary workspace computation algorithms offer an alternative to full workspace algorithms, in particular when we look for better computational performances. As a drawback, these algorithms are less generally applicable than full computation algorithms: holes and voids in the workspace are difficult to identify, and usually the selection of the modelling strategy affects not only the performance of the algorithm, but also their applicability.

1.2 Contributions

In this paper, we propose a boundary workspace computation algorithm for *CPRs*. We aim at reducing the computational time w.r.t. full workspace computation algorithms while preserving their general applicability. Our algorithm is suitable for the boundary computation of any type of planar *CPRs* workspace, but only for translational (i.e. constant orientation) and orientation (i.e. constant position) workspace of spatial *CPRs* [9], since it is based on a three-dimensional grid exploration[1]. To do this, we originally propose i) a suitable exploration strategy for identifying all borders, and ii) a boundary reconstruction algorithm. The exploration strategy is introduced in order to identify a first boundary location, starting from a guess initial point. The exploration strategy is repeated several times in different directions, that depends on the location of previously identified border locations. This way, not only exterior borders but also interior borders and voids (that may occur in the *CPRs* workspace) can be identified efficiently. Then, a boundary reconstruction algorithm is proposed to compute the boundary surface: the border is identified over a fixed grid by exploring the neighborhood of previously identified border points, with the exploration being based on the solution of the *IGSP* over several *EE* locations. In comparison to the state-of-the-art, this algorithm requires reduced computational time w.r.t. full workspace computation algorithms while preserving general applicability.

The proposed algorithm offers several advantages with the respect to the state of the art. First, our algorithm is not related to a specific *CPR* modelling strategy: it requires the use of discretization techniques which encompass a large variety of approaches, as it will be described in Section 2. Consequently, our algorithm may be applied to a broad spectrum of *CPRs*, and a large number of workspace-limiting constraints that may reduce the *CPRs* workspace can be simultaneously considered. Second, to the authors knowledge, no state-of-the-art algorithm for the boundary computation of continuum robots demonstrated the ability to identify holes in the workspace, while our approach does not suffer of this issue. Holes and voids identification is a declared limitation of optimization algorithms [31], and no implementation or simulation results have been shown for continuation approaches [28] concerning hole detection capabilities. Finally, optimization approaches [30] may fail when the robot admits multiple working modes (as we will show later in the case-study section), while this is not the case for the proposed workspace algorithm. Continuation approaches [28] requires the superimposition of several slices for the three-dimensional workspace computations, which may be tricky with complex workspace shapes (as we will show in the case studies). In contrast, our approach overcome this issue by computing directly workspace boundary surface.

The paper is structured as follows. Section 2 recalls a state-of-the-art *CPRs* modelling framework, and it discusses the discretization process, while Section 3 describes in detail the proposed workspace computation algorithm. In Section 4, our approach is applied to four different *CPRs* and compared with state-of-the-art workspace computation algorithms (while maintaining constant the *CPRs* modelling strategy). Section 5 draws conclusions, and Section 6 discusses limitations of our works, and future directions.

2 Modelling

In this Section, we describe the *CPRs* modelling framework employed in this paper. The aim of this Section is to make the paper self-contained and to help the reader in understanding what follows. A brief description of the con-

[1] Reachable, total orientation and dextrous workspace would require six-dimensional grid explorations. We explored three-dimensional task spaces because i) this covers the most frequent cases of interest in parallel robots, ii) the possibility of graphical representation of the results, and iii) to keep the computational cost of the algorithm reasonable. Authors believe that a possible extension of this paper to six-dimensional cases may be developed, but this is outside of the scope of the paper.

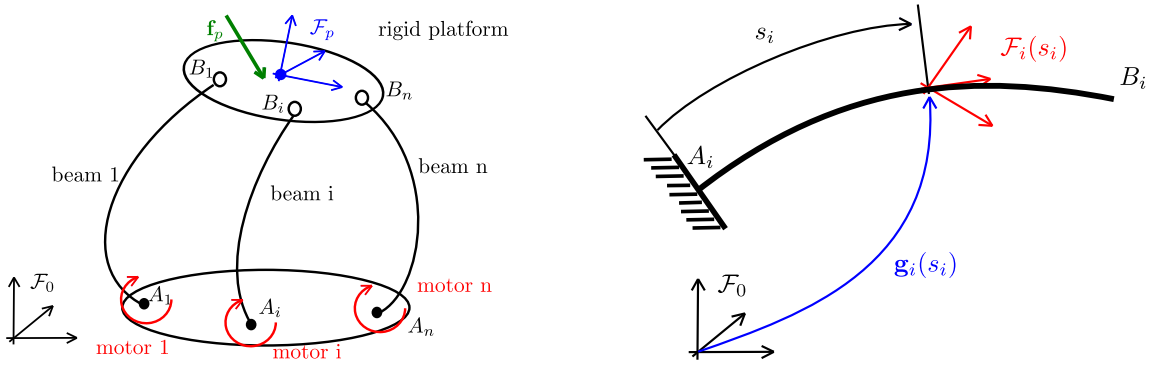


Fig. 1: (a) Schematics of a *CPR*, and (b) continuous parametrization of a slender beam.

sidered *CPRs* architecture is firstly proposed. Then, *CPRs* potential energy is derived (Section 2.1), and the energy-discretization procedure is discussed (Section 2.2) to establish the inverse geometrico-static problem equations (*IGSP*). Finally, equilibrium stability assessment and singularity conditions are discussed in Section 2.3.

Here we consider a *CPR* composed by n continuous beam (called legs, Fig. 1a). Each beam is connected at one extremity (point $A_i, i = 1, \cdots, n$) to a motor and, at the opposite end, to a rigid platform with a passive joint (points B_i)[2]. The robot base frame is denoted with $\mathcal{F}_0$, and a frame $\mathcal{F}_p$ is attached to the rigid platform. The *i*-th actuated variable is called q_{ai}, and the vector $\mathbf{q}_a = [q_{a1}, \cdots, q_{an}] \in \mathbb{R}^n$ collects the actuated variables. The vector $\mathbf{q}_p \in \mathbb{R}^{n_p}$ collects the controlled variables, and we assume $n = n_p$. For instance, for a 6-*DoF* spatial *CPR*, $\mathbf{q}_p = [\mathbf{p}_p, \boldsymbol{\phi}] \in \mathbb{R}^6$ where $\mathbf{p}_p \in \mathbb{R}^3$, $\boldsymbol{\phi} \in \mathbb{R}^3$ represent the position and orientation parameters of the platform frame w.r.t. the fixed frame, respectively.

In the next Sections, we recall the geometrico and kinemato-static modelling and analysis of *CPRs* by using energetic considerations and discretization strategies. Further details can be found in [10], [33].

2.1 *CPR* potential energy

In the static case, and for fixed actuated variables, each beam can be modelled as a clamped beam at the point A_i, as represented in Fig. 1b. The beam is assumed to be initially straight of length L_i and the variable $s_i \in [0, L_i]$ is introduced to represent the curvilinear abscissa of the beam. Moreover, shear and extensibility are assumed to be negligible. A frame $\mathcal{F}_i(s_i)$ is attached at each cross-section of the beam, and the pose of the cross-section is defined by $\mathbf{g}_i(s_i)$ as:

$$\mathbf{g}_i(s_i) = \begin{bmatrix} \mathbf{R}_i(s_i) & \mathbf{p}_i(s_i) \\ \mathbf{0} & 1 \end{bmatrix} \tag{1}$$

where $\mathbf{R}_i(s_i) \in SO(3), \mathbf{p}_i(s_i) \in \mathbb{R}^3$ represent the rotation matrix and the position of the frame $\mathcal{F}_i(s_i)$ w.r.t. $\mathcal{F}_0$, respectively. The curvature and the torsion of the beam expressed in $\mathcal{F}_i(s_i)$ are represented by the vector $\mathbf{u}_i(s_i) \in \mathbb{R}^3$, which can be obtained as [34]:

$$\widehat{\mathbf{u}}_i(s_i) = \mathbf{R}_i^T(s_i)\mathbf{R}_i'(s_i) \tag{2}$$

where $\widehat{\mathbf{u}}_i(s_i) \in so(3)$ is the skew-symmetric matrix obtained from the vector $\mathbf{u}_i(s_i)$, and the superscript $(.)'$ denotes the derivative w.r.t. s_i. The total potential energy of the robot is given by:

$$V_{tot} = V_{plat} + \sum_{i=1}^{n} V_{rod_i} \quad , \quad V_{rod_i} = V_{di} + V_{ei} \tag{3}$$

where $V_{plat} = -\mathbf{f}_p^T \mathbf{p}_p$ represents the potential energy of the rigid platform, $\mathbf{f}_p$ is a conservative force, V_{rod_i} is the total potential energy of the *i*-th beam, V_{ei} the deformation energy, and V_{di} the contribution of distributed loads. Three-dimensional moments, that are non conservative, are considered to not appear [35]. Assuming that only distributed (conservative) forces $\mathbf{f}_i$ act on the beam, V_{di} can be obtained as:

$$V_{di} = -\int_0^L \mathbf{f}_i^T \mathbf{p}_i(s_i) ds \tag{4}$$

Then, we consider shear and extensibility to be negligible (Bernoulli assumptions), material properties as elastic, linear, isotropic, and constant over the beams length. In this case, the deformation energy of the beam is given by [34]:

$$V_{ei} = \int_0^L \left(\mathbf{u}_i(s_i) - \mathbf{u}_i^*(s_i)\right)^T \mathbf{K}_{BT} \left(\mathbf{u}_i(s_i) - \mathbf{u}_i^*(s_i)\right) ds \tag{5}$$

[2] Tendon-actuated parallel *CRs* [21], or *CPRs* with variable length beams [3] could be also modelled. Also, the model is able to take into account passive [6] or intermediate [32] components, and the reader is referred to those papers for additional modelling details hereby not provided for the sake of brevity.

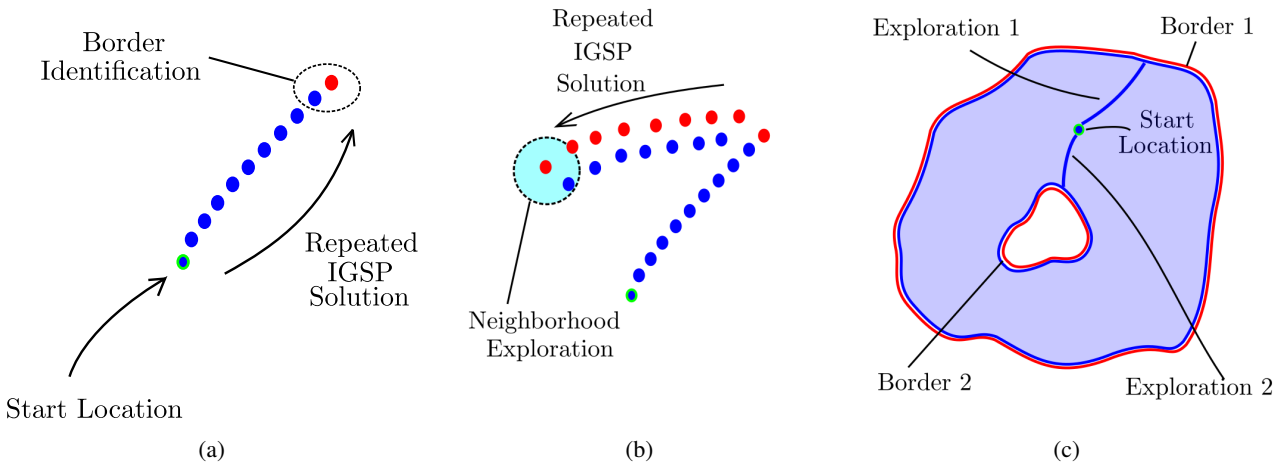


Fig. 2: General overview of the *BFA*. (a) initial grid exploration, (b) boundary computation, and (c) successive iteration of the algorithm. Points on the task space that lie in the workspace are represented in blue, while points outside of the workspace in red.

with $\mathbf{u}_i^*$ is the initial curvature of the beam (null in the case of initially straight beam), $\mathbf{K}_{BT} = \text{diag}(EI_x, EI_y, GI_z)$, E is the Young modulus, G is the shear modulus, and I_x, I_y, I_z are the principal inertia moments of the cross-section.

2.2 Discretized *CPR* equations and equilibrium conditions

CPRs equilibrium configurations are associated with critical points of the robot potential energy V_{tot}, where V_{tot} depends on $\mathbf{q}_a, \mathbf{q}_p$ and on a set of continuous functions $\mathbf{u}_i$. However, finding the values of $\mathbf{q}_a, \mathbf{q}_p$ and the exact functions $\mathbf{u}_i$ that lead to equilibrium conditions of the *CPR* is not trivial. A practical way to solve this problem is to employ discretization strategies. Discretization of the potential energy equations through a finite number of coordinates brings mathematical simplicity where the accuracy of the numerical solution depends on the number of variables, and on the employed discretization strategy [36].

Piecewise constant curvature (*PCC*) modelling approaches enable significant simplification is *CRs* modelling [37]. Being applied successfully in many applications (e.g [38]), *PCC* assumptions brings the advantage of enabling the kinematics to be composed by two simple mappings [39]. However, in case the *CRs* does not have constant curvature shape, modelling inaccuracies may occur. In the direction of developing more accurate *CRs* models, piecewise constant strain (*PCS*) modelling assumptions were introduced [40]. *PCS* introduces the possibility to have torsion, shear and extensibility on the *CRs* model, which enables more complex shapes that a *PCC* model may difficultly represent [41]. However, the number of discretization variables may become high in the case of complex shapes or accurate solutions are required. Variable strain discretization approaches are then proposed [42]: these modelling techniques discretizes the continuum Cosserat beam into a finite set of strain basis functions. Also known as assumed strain modes approaches [43], variable strain discretization approaches bring significant accuracy, but frequently at the cost of high model complexity. Finite differences provide simple and effective expressions for *CRs* models [33], that simplifies robot analysis. These approaches have been applied in several *CRs* applications, such as for concentric tube robot modelling [44] and equilibrium stability analysis [45]. The literature of discretization strategies for *CRs* modelling is significantly large, and finite-element approaches [46], [47], [48], Galerkin-Ritz methods [49], collocation methods [50], [51] also need to be mentioned. However, a deep state-of-the-art analysis of *CRs* modelling technique is out of the scope of this article.

In this paper we employed the assumed strain mode approach of [52] for the simulations proposed in Section 4. We selected this approach because of its good trade-off between accuracy and computational time, and because of the reduced number of elastic variables required to achieve accuracy [53], [54]. However, any other discretization approach could have been used. This approach is mainly based on the discretization of $\mathbf{u}_i$ through base functions as:

$$\mathbf{u}_i(s_i) \simeq \mathbf{N}(s_i)^T \mathbf{q}_{ei} \tag{6}$$

where $\mathbf{N} \in \mathbb{R}^{3\times Nf}$ is a matrix of base functions, N_f is the number of variables that discretizes $\mathbf{u}_i$, and $\mathbf{q}_{ei} \in \mathbb{R}^{N_f}$ is the vector that collects the discretization variables of the *i*-th beam. We assume N_f to be equal for each beam, and the vector $\mathbf{q}_e = [\mathbf{q}_{e1}, \cdots, \mathbf{q}_{en}] \in \mathbb{R}^{nN_f}$ is introduced to collect all the discretization variables.

After introducing the discretization, V_{tot} becomes a function of actuated, platform, and discretization variables, that is $V_{tot} = V_{tot}(\mathbf{q}_a, \mathbf{q}_e, \mathbf{q}_p) = V_{tot}(\mathbf{q}_a, \mathbf{x})$, with $\mathbf{x} = [\mathbf{q}_e, \mathbf{q}_p]$. Due to the closed-loop architecture of *CPRs*, position and orientation geometric constraints have to be enforced. Without loss of generality, the constraints are represented by:

$$\mathbf{\Phi}(\mathbf{q}_a, \mathbf{x}) = \mathbf{0} \tag{7}$$

where a vector $\mathbf{\Phi} \in \mathbb{R}^{n_\Phi}$ is introduced to stack all the geometric constraints in homogeneous form [10], [33].

A robot configuration is an equilibrium configuration if, for fixed values of $\mathbf{q}_a$, $\mathbf{x}$ is a critical point of V_{tot} [55]. However, variables are related by geometric constraints, and critical points of V_{tot} are characterized by Lagrange conditions [55]. Assuming that $\nabla_{\mathbf{x}}\mathbf{\Phi}$ is full rank, $\mathbf{x}$ is a critical point if Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^{n_\Phi}$ exist such as [55]:

$$\begin{cases} \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{q}_a, \mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0} \\ \mathbf{\Phi}(\mathbf{q}_a, \mathbf{x}) = \mathbf{0} \end{cases} \tag{8}$$

with $\mathcal{L} = V_{tot}(\mathbf{q}_a, \mathbf{x}) + \mathbf{\Phi}^T(\mathbf{q}_a, \mathbf{x})\boldsymbol{\lambda}$. Equations (8) represent the implicit geometrico-static model of a *CPR*. The *IGSP* problem requires to find a value of $\mathbf{q}_a$ so that an assigned value of $\mathbf{q}_p^d$ corresponds to a static equilibrium, that is, to solve [33]:

$$\mathbf{F}(\mathbf{y}) = \begin{cases} \nabla_{\mathbf{x}} V_{tot}(\mathbf{q}_a, \mathbf{x}) + \nabla_{\mathbf{x}}\mathbf{\Phi}^T(\mathbf{q}_a, \mathbf{x})\boldsymbol{\lambda} = \mathbf{0} \\ \mathbf{\Phi}(\mathbf{q}_a, \mathbf{x}) = \mathbf{0} \\ \mathbf{q}_p - \mathbf{q}_p^d = \mathbf{0} \end{cases} \tag{9}$$

Equations (9) form a square system of equations of $(nN_f + n) + n_\Phi + n$ equations in $\mathbf{y} = [\mathbf{q}_a, \mathbf{q}_e, \mathbf{q}_p, \boldsymbol{\lambda}]$. Since Eq. (9) is nonlinear, root-finding techniques are employed to identify numerical solutions.

2.3 Equilibrium Configuration Analysis

If a solution of Eq. (9) is found, we want to evaluate if it is feasible for the robot to achieve it: equilibrium stability, singularity conditions, strain limits and joint limits should be considered. To this end, a linear approximation of conventional strain quantities used in beam mechanics is considered [34]:

$$\boldsymbol{\epsilon}(s) = [\gamma_{xz}, \gamma_{yz}, \epsilon_z] = -\mathbf{r}(s) \times \mathbf{u}(s) \tag{10}$$

where $\mathbf{r} = [x_s, y_s, 0]$ is the position of a point that lies over the cross-section in s, and it has coordinate $x_s, y_s, 0$ w.r.t. the local frame $\mathcal{F}(s)$. γ_{xz}, γ_{yz} represent shear strains approximation, and ϵ_z is the normal strain. Strain limits are considered by checking if each of the *CPRs* beam does not exceed the material limits. Actuator limits are verified if:

$$q_{ai} \in [q_{i-min}, q_{i-max}] \quad , \quad i = 1, \cdots, n \tag{11}$$

and q_{i-min}, q_{i-max} are the minimum and maximum position or orientation limits of the *i*-th actuator[3]. Motor efforts τ_i (such as torques for revolute actuators) can be recovered as [10]:

$$\tau_i = \nabla_{q_{ai}} V_{tot}(\mathbf{q}_a, \mathbf{x}) + \nabla_{q_{ai}}\mathbf{\Phi}^T(\mathbf{q}_a, \mathbf{x})\boldsymbol{\lambda} \tag{12}$$

Then, to analyze equilibrium stability and singularity conditions, the Jacobian matrix $\mathbf{J}$ of the *IGSP* equations is computed as [22]:

$$\mathbf{J} = \frac{\partial \mathbf{F}}{\partial \mathbf{y}} = \begin{bmatrix} \mathbf{A}_\mathcal{L} & \mathbf{U}_\mathcal{L} & \mathbf{P}_\mathcal{L} & \mathbf{\Lambda}_\mathcal{L} \\ \mathbf{A}_\mathbf{\Phi} & \mathbf{U}_\mathbf{\Phi} & \mathbf{P}_\mathbf{\Phi} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \end{bmatrix} \tag{13}$$

where:

1. $\mathbf{A}_\mathcal{L} = \nabla_{\mathbf{q}_a}(\nabla_{\mathbf{x}}\mathcal{L})$, $\mathbf{U}_\mathcal{L} = \nabla_{\mathbf{q_e}}(\nabla_{\mathbf{x}}\mathcal{L})$
2. $\mathbf{P}_\mathcal{L} = \nabla_{\mathbf{q}_P}(\nabla_{\mathbf{x}}\mathcal{L})$, $\mathbf{\Lambda}_\mathcal{L} = \nabla_{\boldsymbol{\lambda}}(\nabla_{\mathbf{x}}\mathcal{L})$
3. $\mathbf{A}_\Phi = \nabla_{\mathbf{q}_a}\mathbf{\Phi}$, $\mathbf{U}_\Phi = \nabla_{\mathbf{q}_e}\mathbf{\Phi}$, $\mathbf{P}_\Phi = \nabla_{\mathbf{q}_P}\mathbf{\Phi}$

Singularity conditions occurs by the degeneracy of the following matrices [10]:

$$\mathbf{T}_1 = \begin{bmatrix} \mathbf{Z}^T\mathbf{A}_\mathcal{L} & \mathbf{Z}^T\mathbf{U}_\mathcal{L} \\ \mathbf{A}_\mathbf{\Phi} & \mathbf{U}_\mathbf{\Phi} \end{bmatrix} \tag{14}$$

$$\mathbf{T}_2 = \begin{bmatrix} \mathbf{Z}^T\mathbf{P}_\mathcal{L} & \mathbf{Z}^T\mathbf{U}_\mathcal{L} \\ \mathbf{P}_\mathbf{\Phi} & \mathbf{U}_\mathbf{\Phi} \end{bmatrix} \tag{15}$$

where $\mathbf{Z}$ spans the left nullspace of $\mathbf{\Lambda}_\mathcal{L}$, i.e $\mathbf{Z}^T\mathbf{\Lambda}_\mathcal{L} = \mathbf{0}$. While the degeneracy of $\mathbf{T}_1$ is associated with serial (or Type 1) singularities, rank deficiency of $\mathbf{T}_2$ is related to parallel (or Type 2) singularities. Then, under the assumption that only external conservative loads act on the robots platform, equilibrium stability can be assessed by evaluating the positive definiteness of the reduced Hessian matrix $\mathbf{H}^r$ of the total potential energy[4] [33] , [55] that is:

$$\mathbf{H}^r = \mathbf{Z}^T\frac{\partial^2\mathcal{L}}{\partial\mathbf{x}^2}\mathbf{Z} = \mathbf{Z}^T\begin{bmatrix}\mathbf{U}_\mathcal{L} & \mathbf{P}_\mathcal{L}\end{bmatrix}\mathbf{Z} \tag{16}$$

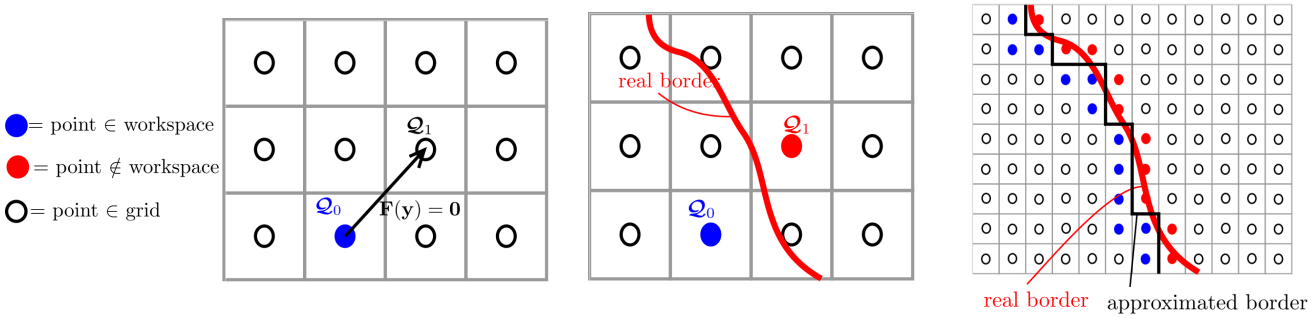
3 Workspace Algorithm

This Section describes the workspace computation methodology introduced in this paper, called boundary flooding algorithm (*BFA*). We first give a general overview of *BFA* to ease the reader's comprehension. Then, boundary identification is discussed in Sec. 3.1, and details of the algorithm are given in Sec. 3.2.

Similarly to conventional discretization approaches (e.g [19]), our algorithm is a task-space discretization algorithm, and a grid of dimension at most three is generated to discretize the task-space of the robot. The *IGSP* of *CPRs*

[3] In a similar fashion, passive joint range limits can be considered by recovering their values from the *IGSP* solution, and by verifying if the joint limits are respected

[4] The notation $\mathbf{H}^r$ is used in accordance with [10], [33] to distinguish between the reduced Hessian matrix, and the Hessian matrix $\mathbf{H}$ of the total potential energy.



(a) (b) (c)

Fig. 3: Boundary identification over a discrete grid. (a) *IGSP* solution over the grid, (b) border identification, and (c) border reconstruction.

Algorithm 1: Boundary flooding algorithm.

Initialize grid, toDo, $\mathcal{Q}_{start}$, $\mathcal{A}$, $\mathcal{B}$;
for k = *1:* n_{exp} **do**
  toDo = Space Exploration($\mathcal{Q}_{Ak}$, $\mathcal{B}$, $\mathcal{Q}_{start}$);
  Boundary Computation(toDo);
end
Function `Space Exploration` ($\mathcal{Q}_{Ak}$, $\mathcal{B}$, $\mathcal{Q}_{start}$)**:**
  $\mathcal{Q}_{init} = \mathcal{Q}_{start}$;
  while *flag* = *true* **do**
    $\mathcal{Q}_{end}$ = getNewPoint($\mathcal{Q}_{init}$, $\mathcal{Q}_{Ak}$, $\mathcal{B}$, iter);
    flag = GetWKconditions($\mathcal{Q}_{init}$, $\mathcal{Q}_{end}$);
    if *flag* = *true* **then**
      Save Results;
      iter = iter + 1;
      $\mathcal{Q}_{init} = \mathcal{Q}_{end}$;
    end
  end
  Stack [$\mathcal{Q}_{init}$, $\mathcal{Q}_{end}$] ∈ toDo;
  Stack $\mathcal{Q}_{end}$ in $\mathcal{B}$;
  return
Function `Boundary Computation` (*toDo*)**:**
  while *toDo* $\neq \emptyset$ **do**
    Extract $\mathcal{Q}_g$, $\mathcal{Q}_i$ from *toDo*;
    $\mathcal{N}_i$ = get neighbors of $\mathcal{Q}_i$ not yet computed.;
    Sort $\mathcal{N}_i$ w.r.t. distance from $\mathcal{Q}_g$;
    while $\mathcal{N}_i \neq \emptyset$ **do**
      $\mathcal{Q}_{n1} = \mathcal{N}_i(1)$;
      $\mathcal{N}_i \leftarrow \mathcal{N}_i \backslash \mathcal{Q}_1$;
      $\mathcal{N}_{w1}$ = get neighbors of $\mathcal{Q}_1$ ∈ WK;
      if $\mathcal{N}_{w1} \neq \emptyset$ **then**
        $\mathcal{Q}_w = \mathcal{N}_{w1}$ with best conditioning of $\mathbf{J}$;
        flag = GetWKconditions($\mathcal{Q}_w$, $\mathcal{Q}_{n1}$);
        if *flag* = *true* **then**
          Save Results.
        else
          Stack [$\mathcal{Q}_w$, $\mathcal{Q}_{n1}$] ∈ toDo
        end
      end
    end
  end
  return
Function `GetWKConditions` ($\mathcal{Q}_0$, $\mathcal{Q}_1$)**:**
  $\mathbf{y}_0$ = robot configuration at $\mathcal{Q}_0$;
  $\mathbf{y}$ = Solve IGSP starting from $\mathbf{y}_0$;
  $flag$ = evaluate if $\mathcal{Q}_1$ ∈ WK;
  return $flag$;

is solved repeatedly over the grid, but not on all its locations, since the goal of the algorithm is to reconstruct the workspace boundaries only. We decided to investigate at most three-dimensional task-spaces, thus excluding reachable, total orientation and dextrous workspace of spatial *CPRs*, because i) this covers the most frequent cases of interest in parallel manipulators, ii) the possibility of graphical visualization of the results, iii) to keep the computational cost of the algorithms reasonable, and iv) to compare with state-of-the-art-results. Authors believe that a possible extension of the *BFA* to six-dimensional cases may be developed, but this is outside of the scope of this paper. The algorithm is designed as a two-sequential-stage process:

1. starting from a known location, the grid is explored by repeatedly solving the *IGSP* over different points until a border point of the workspace is reached, as shown in Fig. 2a. This stage is described in detail in Sec. 3.2.1;
2. then, the border is computed thanks to a flooding algorithm specifically designed for this scope. This algorithm is based on the repeated solution of the *IGSP* (Fig. 2b), and the determination of new points to explore in the neighborhood of previously identified borders. This stage is illustrated in Sec. 3.2.2

These two stages can be repeated several times: successive explorations directions are pointed toward different regions of the grid to possibly identify holes and voids that the *CPRs* workspace may possess (Fig. 2c). The exploration strategy takes advantage of previously detected borders to scan only regions on the grid where no borders were identified.

3.1 Boundary Identification

A crucial point of the *BFA* is how boundaries are identified. As shown in Fig. 3a, points are placed at the center of each box of the grid. Then, we assume that the point $\mathcal{Q}_0$ lies in the workspace and we want to verify whether a neighbor point $\mathcal{Q}_1$ is included in the workspace as well. To do that, being $\mathbf{y}_0$ the robot configuration at $\mathcal{Q}_0$, the *IGSP* is solved with initial guess $\mathbf{y}_0$, and the resulting configuration $\mathbf{y}_1$ is obtained. Then, we check if $\mathcal{Q}_1$ lies in the workspace: this is done by verifying:

1. singularity conditions, identified by the conditions reported in Eq. (15);
2. equilibrium stability conditions, verified thanks to Eq. (16);
3. strain limits and actuator limits, that can be recovered from Eq. (10), Eq. (11), Eq. (12).

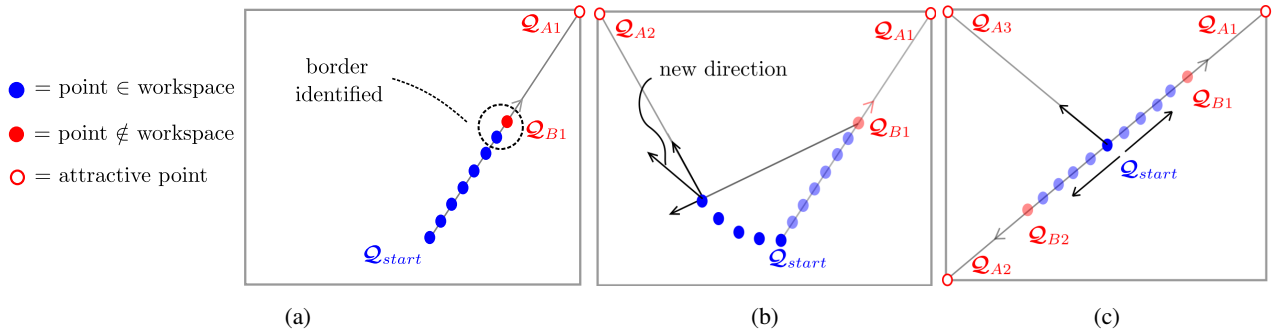


Fig. 4: Exploration strategy: (a) first exploration, (b) subsequent exploration and influence of previously computed points, (c) conditions for which an equilibrium is reached without attractive points.

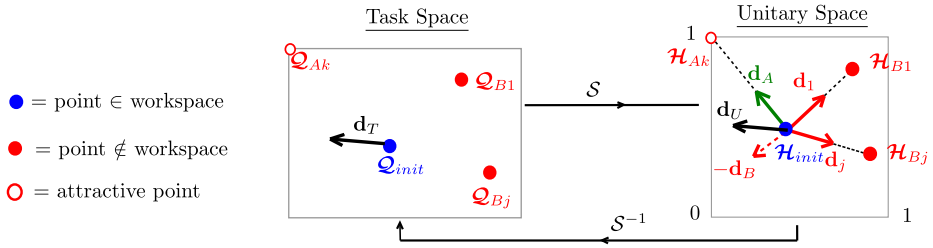


Fig. 5: Space exploration strategy. On the left, points on the physical space. On the middle, points converted into the unitary space. On the right, computation of the new exploration direction.

Other conditions may delimit the robot workspace, and general inequalities (and equalities) can be considered, but we limit our analysis to the aforementioned criterions.

Then, if the configuration $\mathbf{y}_1$ violates one of the aforementioned conditions, the boundary of the workspace is crossed, and $\boldsymbol{\mathcal{Q}}_1$ is considered as an out-of-the-workspace point. Since the task-space is discretized with a grid, the real boundary is placed in a location between $\boldsymbol{\mathcal{Q}}_0$ and $\boldsymbol{\mathcal{Q}}_1$, as represented in Fig. 3b. Thus, the boundary can be approximately reconstructed as shown in Fig. 3c and the accuracy of the workspace estimation depends on the grid sampling size.

3.2 Detailed Description

In this section, we describe in detail the workspace computation algorithm objective of this paper. We firstly illustrate the space exploration strategy in Sec. 3.2.1, and then the boundary computation strategy is proposed in Sec. 3.2.2.

3.2.1 Space Exploration Strategy

The main goal of the space exploration strategy (Alg. 1, lines 6-19) is to explore the grid and to identify a point that lies on the workspace boundary. As previously mentioned, the exploration attempts to scan unexplored grid regions by considering where previous explorations were directed. In order to explain how the exploration directions are obtained, we need to introduce two different concepts: the unitary space, and the attractive points.

Unitary Space: since we seek to explore task spaces that may involve positions and orientations at the same time, the computation of the exploration direction is performed on a unitary space (Fig. 5), that is, a space with $[0, 1]$ limits at each direction algebraically similar to the euclidean space. Given a generic point in the task-space $\boldsymbol{\mathcal{Q}}$, its similarity transformation $\mathcal{S}$ into the unitary space point $\boldsymbol{\mathcal{H}}$ is given by:

$$\boldsymbol{\mathcal{H}} = \mathcal{S}(\boldsymbol{\mathcal{Q}}) \tag{17}$$

As an example, being $\boldsymbol{\mathcal{Q}} = [x, y, z]$, and $x_{lim} = [a, b]$, $y_{lim} = [c, d]$, $z_{lim} = [e, f]$ the task-space limits, the point $\boldsymbol{\mathcal{H}}$ is obtained as:

$$\boldsymbol{\mathcal{H}} = \left(\frac{x-a}{b-a}, \frac{y-c}{d-c}, \frac{z-e}{e-f} \right)^T \tag{18}$$

Attractive Points: these points are introduced in order to define exploration directions, and they are positioned at the limits of the grid. These grid limits are usually placed at regions not reachable by the manipulator (e.g. at a distance longer than the robot leg lengths), and attractive points are placed uniformly over the grid perimeter. Being n_{exp} the total number of explorations, $\boldsymbol{\mathcal{A}} = [\boldsymbol{\mathcal{Q}}_{A1}, \cdots, \boldsymbol{\mathcal{Q}}_{An_{exp}}]$ collects the attractive points.

After unitary space and attractive points are introduced, let us explain how the exploration works. To do that, let us consider a task-point $\boldsymbol{\mathcal{Q}}_{start}$ that lies in the workspace where the exploration starts[5]. We seek to solve the *IGSP* over differ-

[5]The identification of the first point is not trivial, since *IGSP* Eqs. (9) are nonlinear. An efficient heuristic employs constant curvature assumptions [21], where the inverse problems admits simple (and possibly not accurate) purely geometric solutions. These constant curvature solutions are employed as initial guess for the first solution of Eqs. (9).

ent grid locations until a border of the workspace is reached. Thus, we need to sequentially i) solve the *IGSP* and ii) find a new task-space point where to solve the *IGSP*. The latter is the key point of the exploration strategy: at each step, we need to find an exploration direction on the task space that defines the new *IGSP* point by taking into account the previous explorations (to not explore the same grid regions).

During the first exploration ($k = 1$ with $k = 1, \cdots, n_{exp}$ the index representing the exploration number) the grid is explored by solving the *IGSP* over sequential new locations in the direction of $\boldsymbol{\mathcal{Q}}_{A1}$ until a border is identified (Fig. 4a). Being $\boldsymbol{\mathcal{Q}}_{B1}$ the point where the border is identified, the exploration is then stopped and $\boldsymbol{\mathcal{Q}}_{B1}$ is stored in $\boldsymbol{\mathcal{B}}$, which is the set of all the border points. Then, for the second exploration, we consider $\boldsymbol{\mathcal{Q}}_{A2}$ and the exploration restart from $\boldsymbol{\mathcal{Q}}_{start}$. However, we would like to consider also the influence of $\boldsymbol{\mathcal{Q}}_{B1}$ and the exploration should be directed also in a direction opposite to $\boldsymbol{\mathcal{Q}}_{B1}$ to explore a different region than the previous one. Therefore, the exploration direction is obtained as a combination of the direction that points toward $\boldsymbol{\mathcal{Q}}_{A2}$, and the direction opposite to $\boldsymbol{\mathcal{Q}}_{B1}$, as qualitatively represented in Fig. 4b.

To obtain the mathematical expression of the exploration direction, let us consider the generic *k*-th exploration, where $\boldsymbol{\mathcal{B}} = [\boldsymbol{\mathcal{Q}}_{B1}, \cdots, \boldsymbol{\mathcal{Q}}_{B(k-1)}]$ collects out-of-the-workspace point locations identified in the previous $k-1$ iterations. During the *k*-th exploration, $\boldsymbol{\mathcal{Q}}_{init}$ represents the current workspace point and we seek to select a new point in its neighborhood where to solve the *IGSP*, with initial guess $\boldsymbol{\mathcal{Q}}_{init}$. Thus, we need to identify a direction $\mathbf{d}_T$ in the task space that is used to select the next *IGSP* point. To do that, we employ the unitary space: all the points of interest in the task-space are mapped into the unitary space by employing the transformation $\mathcal{S}$, and $\boldsymbol{\mathcal{H}}_{init}, \boldsymbol{\mathcal{H}}_{Ak}, \boldsymbol{\mathcal{H}}_{B1}, \cdots, \boldsymbol{\mathcal{H}}_{B(k-1)}$ are the unitary space counterparts of $\boldsymbol{\mathcal{Q}}_{init}, \boldsymbol{\mathcal{Q}}_{Ak}, \boldsymbol{\mathcal{Q}}_{B1}, \cdots, \boldsymbol{\mathcal{Q}}_{B(k-1)}$ (Fig. 4c). Then, the exploration direction is given by:

$$\mathbf{d}_T = \mathcal{S}^{-1}(\mathbf{d}_U) \quad , \quad \mathbf{d}_U = \frac{c_A \mathbf{d}_A - c_B \mathbf{d}_B}{\|c_A \mathbf{d}_A - c_B \mathbf{d}_B\|} \tag{19}$$

with $\mathbf{d}_U$ the exploration direction in the unitary space (Fig. 5), $\mathbf{d}_A$ the unitary vector defined by the *k*-th attractive point, $\mathbf{d}_B$ the unitary vector defined by the $k-1$ border points, $c_A = 1 - exp(-iter/\tau)$, $c_B = exp(-iter/\tau)$, $iter$ being the cumulative number of *IGSP* solved during the *k*-th exploration, and τ a constant that defines the behaviour of the exploration. This way, at the start of the exploration, the influence of previous computation is relevant, and only after several *IGSP* solutions the direction is pointed mainly toward the attractive point.

We now detail the terms of Eq. (19). Attractive point direction $\mathbf{d}_A$ is given by:

$$\mathbf{d}_A = \frac{\boldsymbol{\mathcal{H}}_{init} - \boldsymbol{\mathcal{H}}_{Ak}}{\|\boldsymbol{\mathcal{H}}_{init} - \boldsymbol{\mathcal{H}}_{Ak}\|} \tag{20}$$

Then, for the computation of $\mathbf{d}_B$ we propose the following heuristic:

$$\mathbf{d}_B = \frac{\sum_{j=1}^{N_b} c_{Bj} \mathbf{d}_{Bj}}{\| \sum_{j=1}^{N_b} c_{Bj} \mathbf{d}_{Bj} \|}, \quad \mathbf{d}_{Bj} = \frac{\boldsymbol{\mathcal{H}}_{init} - \boldsymbol{\mathcal{H}}_{Bj}}{\|\boldsymbol{\mathcal{H}}_{init} - \boldsymbol{\mathcal{H}}_{Bj}\|} \tag{21}$$

Authors experienced good results by employing this heuristic during the simulations of Sec. 4. From Eq. (21) we can see that $\mathbf{d}_B$ is a weighted sum of the directions $\mathbf{d}_{B_j}$ defined by each border point, with weights c_{Bj} defined as:

$$c_{Bj} = \|\mathbf{1} - (\boldsymbol{\mathcal{H}}_{init} - \boldsymbol{\mathcal{H}}_{Bj})\| \tag{22}$$

The meaning of the coefficient c_{Bj} is not trivial: since we operates on the unitary space, each components of $(\boldsymbol{\mathcal{H}}_{init} - \boldsymbol{\mathcal{H}}_{Bj})$ is bounded between $0, 1$. Thus, when $\boldsymbol{\mathcal{H}}_{init}$ approaches $\boldsymbol{\mathcal{H}}_{Bj}$ the value of c_{Bj} increases and $\boldsymbol{\mathcal{H}}_{Bj}$ have a larger influence on the calculation of $\mathbf{d}_B$. This way, if the exploration approaches a previously identified border, its weight on $\mathbf{d}_B$ increases and $\mathbf{d}_U$ is modified accordingly.

To resume, at each step of the exploration *k*-th , we proceed in this way:

1. all the point of interest (attractive point $\boldsymbol{\mathcal{Q}}_{Ak}$, current workspace point $\boldsymbol{\mathcal{Q}}_{init}$, and border points $\boldsymbol{\mathcal{B}}$), are mapped into the unitary space;
2. then, the exploration direction is computed into the unitary space and mapped back into the task space by employing Eq. (19);
3. given the exploration direction $\mathbf{d}_T$, the neighbor of $\boldsymbol{\mathcal{Q}}_{init}$ that points in the direction closer to $\mathbf{d}_T$ is selected as new point. This task-space point is called $\boldsymbol{\mathcal{Q}}_{end}$.

These three steps are performed at line 9 of Alg. 1. Then, the *IGSP* is solved and we check if $\mathbf{Q}_{end}$ lies on the workspace. The computation continues point-by-point until a border is identified. Once the boundary is found, $\boldsymbol{\mathcal{Q}}_{Bk}$ is saved in $\boldsymbol{\mathcal{B}}$ for next explorations, and $\boldsymbol{\mathcal{Q}}_{init}, \boldsymbol{\mathcal{Q}}_{end}$ are stored in *toDo* for the boundary computation phase described in Sec. 3.2.2. Before going to the description of the boundary computation strategy, we need to remark few details:

- *Necessity of attractive points:* even if attractive points seem to be unnecessary after the first exploration (the exploration direction could be defined by points in $\boldsymbol{\mathcal{B}}$ only), it can happen that previously computed borders define directions for which the computation stalls. An example is reported in Fig. 4c: since $\boldsymbol{\mathcal{Q}}_{start}$ is placed in the middle of the two border points $\boldsymbol{\mathcal{Q}}_{B1}, \boldsymbol{\mathcal{Q}}_{B2}$, the blocked direction $\mathbf{d}_B$ of Eq. (21) is indeterminate. Thus, attractive points are required to ensure the algorithms does not stall, and to drive the exploration away from the stall condition.
- *Exploration parameter.* The coefficient τ is user-defined and it should be selected in relation to the scope of the workspace exploration. An high value of τ causes more complex exploration path, at the cost of more iterations. On the other hand, a reduced value of τ should

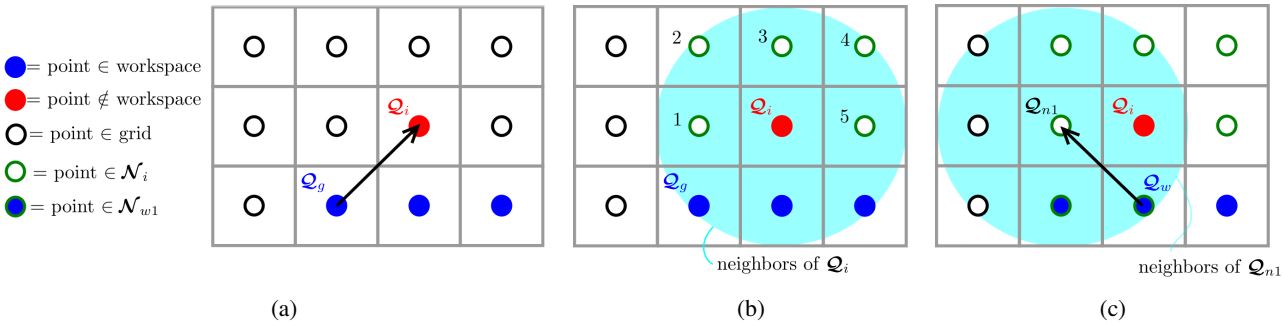


Fig. 6: Boundary computation strategy: (a) boundary conditions, (b) sorting strategy, (c) point evaluation.

be employed when the user has a previous knowledge of the workspace shape (e.g. slight robot design modifications), the exploration is mainly pointed toward attractive points placed at location useful to identify all the workspace boundary components. Typical values of τ are related to the grid sampling size s_g, and the grid size. Empirically, the authors experienced optimal results when initializing τ so that $s_g\tau$ is approximately half of the grid size.

- *Number of explorations.* n_{exp} is a user-defined parameter and it depends on the scope of the workspace exploration. In the case no previous knowledge of the workspace shape is available, the authors experienced optimal results with high values of n_{exp} ($n_{exp} \geq 20$); even though this number is quite high and results in a higher computational time than needed most of the times, it gives higher assurance to find all the workspace borders sought. However, if the algorithm is supposed to be used several times (e.g. when performing slight design variations), n_{exp} should and could be reduced to save computational time, since there is a-priori knowledge of the workspace borders[6].

3.2.2 Boundary Computation Strategy

In this subsection, the algorithm for the reconstruction of the workspace boundaries is detailed. The pseudocode of this routine is reported in Alg. 1, lines. 20-38. The algorithm starts from a situation where a point $\mathcal{Q}_i$ outside the boundary is identified after the solution of the *IGSP*, with initial guess $\mathcal{Q}_g$ (Fig. 6a). These two points are extracted from *toDo*, an array that stores out-of-the-workspace points and their previously employed initial guesses. Subsequently, the goal is to explore the grid in order to identify new points outside the workspace, and to reconstruct the complete boundary. To do that, we identify neighbors points to $\mathcal{Q}_i$ over the grid, and we store in $\mathcal{N}_i$ the neighbors where the *IGSP* has not been computed (Fig. 6b).

Points in $\mathcal{N}_i$ represent possible candidate to be out-of-the-workspace points: in order to verify whether they lie or not in the workspace, the *IGSP* should be solved and an appropriate initial guess should be identified for each point. However, the order on which points in $\mathcal{N}_i$ are treated may influence the resulting prediction, since a point may have an appropriate initial guess that is not yet found during the workspace computation. In this work, we propose to test points $\mathcal{N}_i$ w.r.t. to their distance from $\mathcal{Q}_g$, starting from the closer[7] (see Fig. 6b). Authors experienced good results with this approach, but other heuristics may be proposed.

The next step requires to solve the *IGSP* for all the points in $\mathcal{N}_i$. Being $\mathcal{Q}_{n1}$ the first point to be computed in $\mathcal{N}_i$, we seek to solve the *IGSP*, and an initial guess should be identified. To do that, neighbor points of $\mathcal{Q}_{n1}$ that lie in the workspace are collected in $\mathcal{N}_{w_1}$ (Fig. 6c). In case $\mathcal{N}_{w_1}$ is not empty[8], the point with the best conditioning of $\mathbf{J}$ (Eq. (13)) is extracted among the possible initial guesses in $\mathcal{N}_{w_1}$, and named as $\mathcal{Q}_w$ (Fig. 6c). By selecting the initial guess with this heuristic, we obtain high probability that the new configuration preserves the working mode of the initial guess, but no analytical proof is available. In the case the preservation of the working mode becomes important, authors previous work can be employed [22]. Finally, we evaluate if $\mathcal{Q}_{n1}$ lies on the workspace. In case $\mathcal{Q}_{n1}$ is included in the robot workspace, no additional points are stored in *toDo*. On the opposite case, $\mathcal{Q}_w$, $\mathcal{Q}_{n1}$ are stored in *toDo* for next computation.

The algorithm continues by at first evaluating all the points in $\mathcal{N}_i$ and solving the *IGSP* at all these points. Once $\mathcal{N}_i$ is empty, new points are extracted from *toDo*. When also *toDo* is empty, the algorithm stops. In the next Section, we apply the proposed algorithm for the workspace computation of four different *CPRs*.

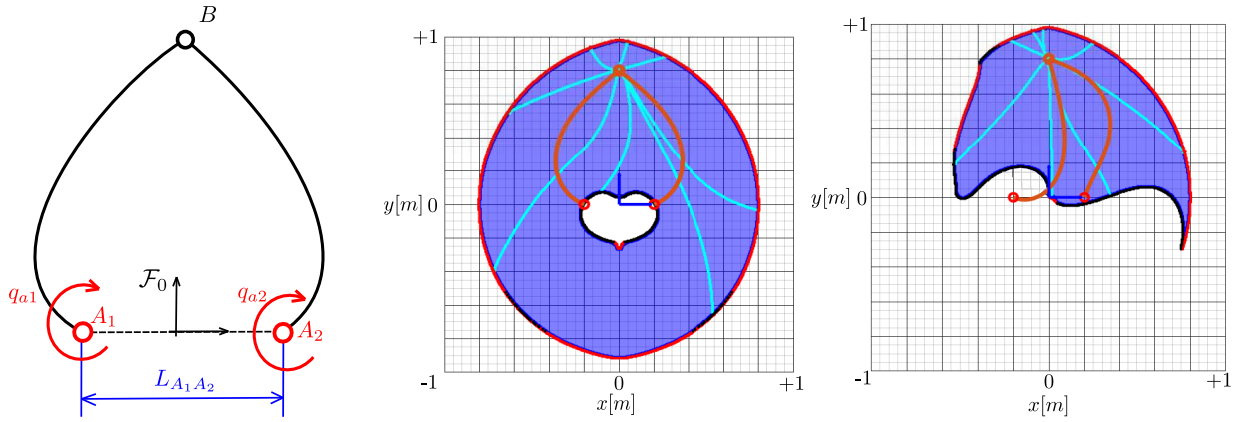
4 Case Studies

In this Section, we propose four case studies to show the main features of the proposed workspace border computation algorithm. We compare two workspace algorithms with our approach to validate applicability of our method, highlight its merits, and also show its main limitations. We compare our approach with the full workspace algorithm [10] that supports three-dimensional investigations (in contrast to our previous work [22] that focuses on planar cases only),

[6] Some workspace slices can be computed *a priori* with the algorithm of [10], or [22], to detect the presence of eventual holes, and then n_{exp} adjusted consequently

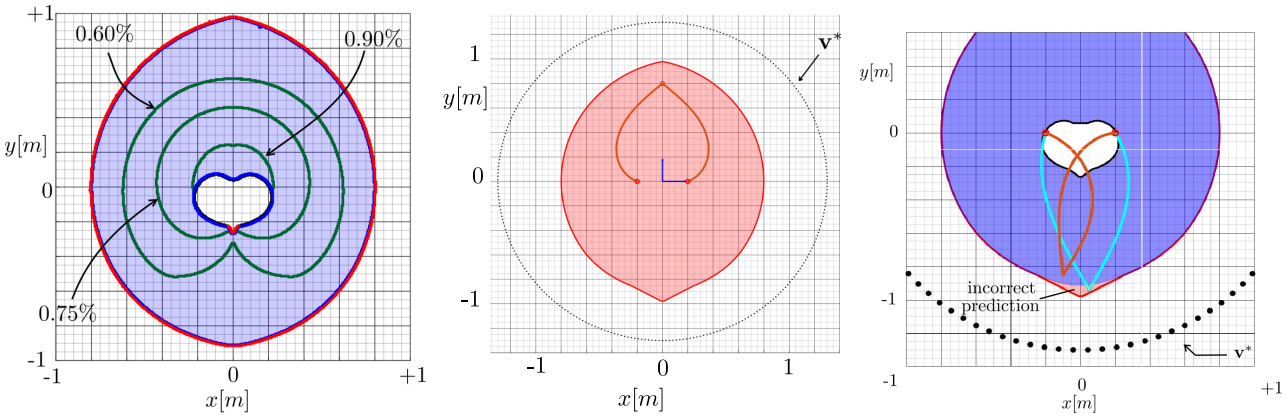
[7] In case points have the same distance, we randomly selected one.

[8] In case $\mathcal{N}_{w_1} = \emptyset$, the point $\mathcal{Q}_{n1}$ is skipped, and the *IGSP* not solved.



(a) RFRFR robot (b) RFRFR robot working mode 1 (c) RFRFR robot working mode 2

Fig. 7: Workspace computation of the $\underline{R}FRF\underline{R}$ robot: the robot architecture (a), and the workspace computation of two different working modes (b),(c). Exploration trajectories are depicted in light blue, results obtained with the approach of [10] in dark blue, Type-1 and Type-2 singularities are represented in red, and black, respectively.



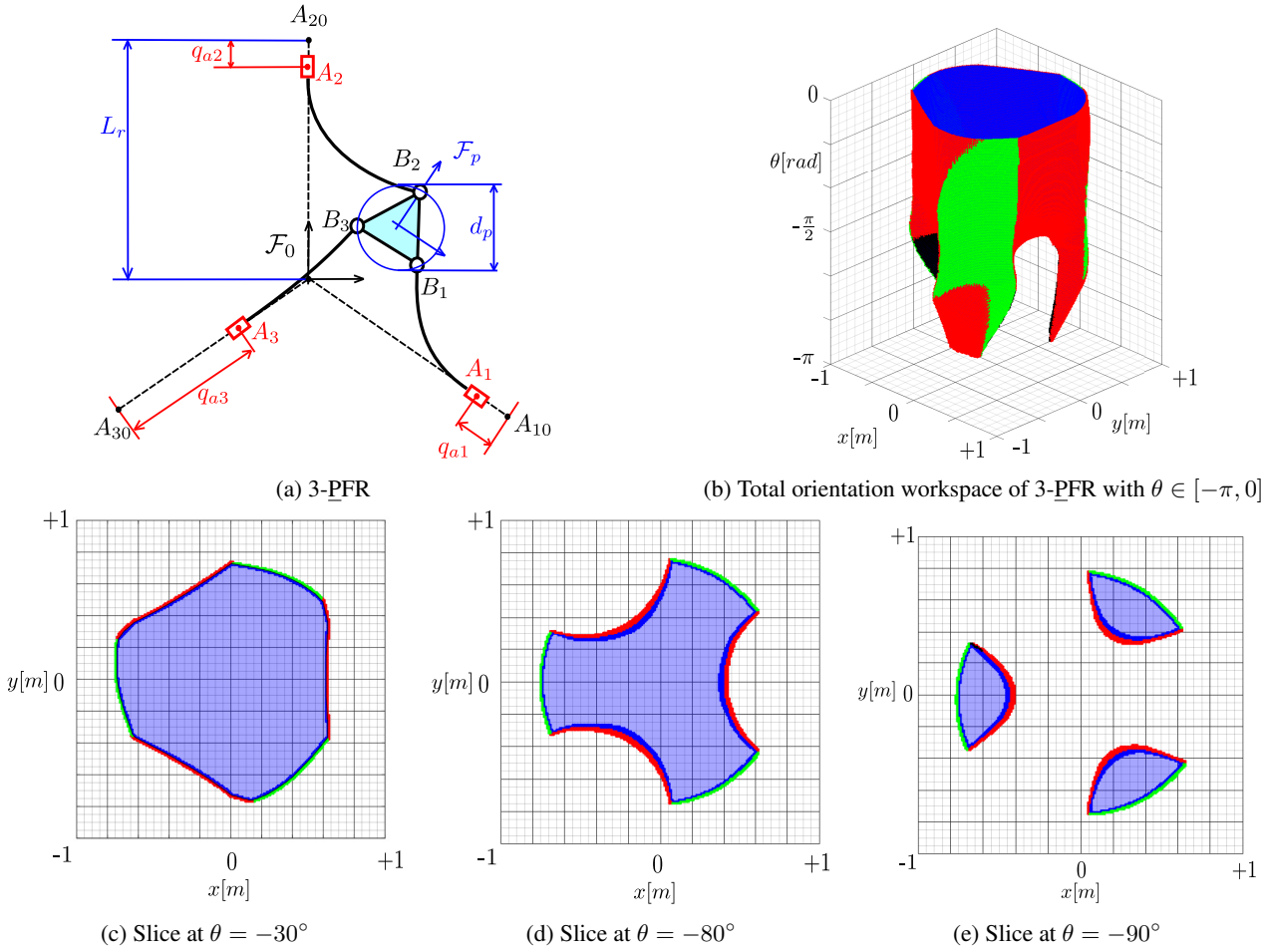
(a) Strain investigation (b) Results with [30] (c) Comparison

Fig. 8: Workspace computation of the $\underline{R}FRF\underline{R}$ robot. Influence of strain limits on the robot workspace (a), with in green the inner borders with different strain limits. In (b), the results obtained with the approach of [30]. In (c), comparison on the external border prediction between our approach and the approach of [30].

to verify the correctness of the results. Also, we compare the *BFA* with the boundary computation algorithm of [30], since it presents interested features in terms of computational time and, even if it is designed for serial *CRs*, it can be used also for *CPRs*. However, we keep the same modelling strategy detailed in Section. 2 for each workspace algorithm, and we selected $N_f = 4$ in Eqs. (6) ($\mathbf{u}$ is approximated with four orthogonal Legendre polynomials). For all case studies, beams are made of harmonic steel with Young modulus $E = 210$ GPa, beam length is equal to $L = 1$ m. Beams are of circular cross-section with radius $r = 1$ mm. Simulations of this Section are performed with a PC equipped with a CPU Intel Core i7-6700, 3.4GHz, 32Gb RAM, in a Matlab environment. The *IGSP* is solved by Matlab *fsolve* routine, and equations are precompiled as *.mex* function to speed up the computation. A trust-region algorithm is selected since it required less computational time than other available algorithms (e.g. Levenberg-Marquardt), and the maximum allowed iteration number for the trust-region algorithm is set as 20. If the number of iterations reaches 20, we consider that no solution to the *IGSP* exists. Despite a reduced number of iterations is usually required in regions not close to the border (e.g 4-5 iterations), the *IGSP* solution in regions near workspace boundaries related to singularities of the *IGSP* problem (Type-1 singularities [10]) usually requires numerous iterations that could slow down the algorithm. In this way, Type-1 singularities can be detected more rapidly.

4.1 RFRFR robot

The $\underline{R}FRF\underline{R}$ robot has been introduced in [18], and its workspace computation was investigated in our previous work [22]. The $\underline{R}FRF\underline{R}$ robot has two revolute actuators ($\underline{R}$) placed at the base (points A_1, and A_2 in Fig. 7a) that rotate the base of two flexible beams (F). Beams are connected at the opposite side through a passive revolute joint (R). For this case study, the distance between the motors $L_{A_1A_2}$ is



(a) 3-PFR

(b) Total orientation workspace of 3-PFR with $\theta \in [-\pi, 0]$

(c) Slice at $\theta = -30°$

(d) Slice at $\theta = -80°$

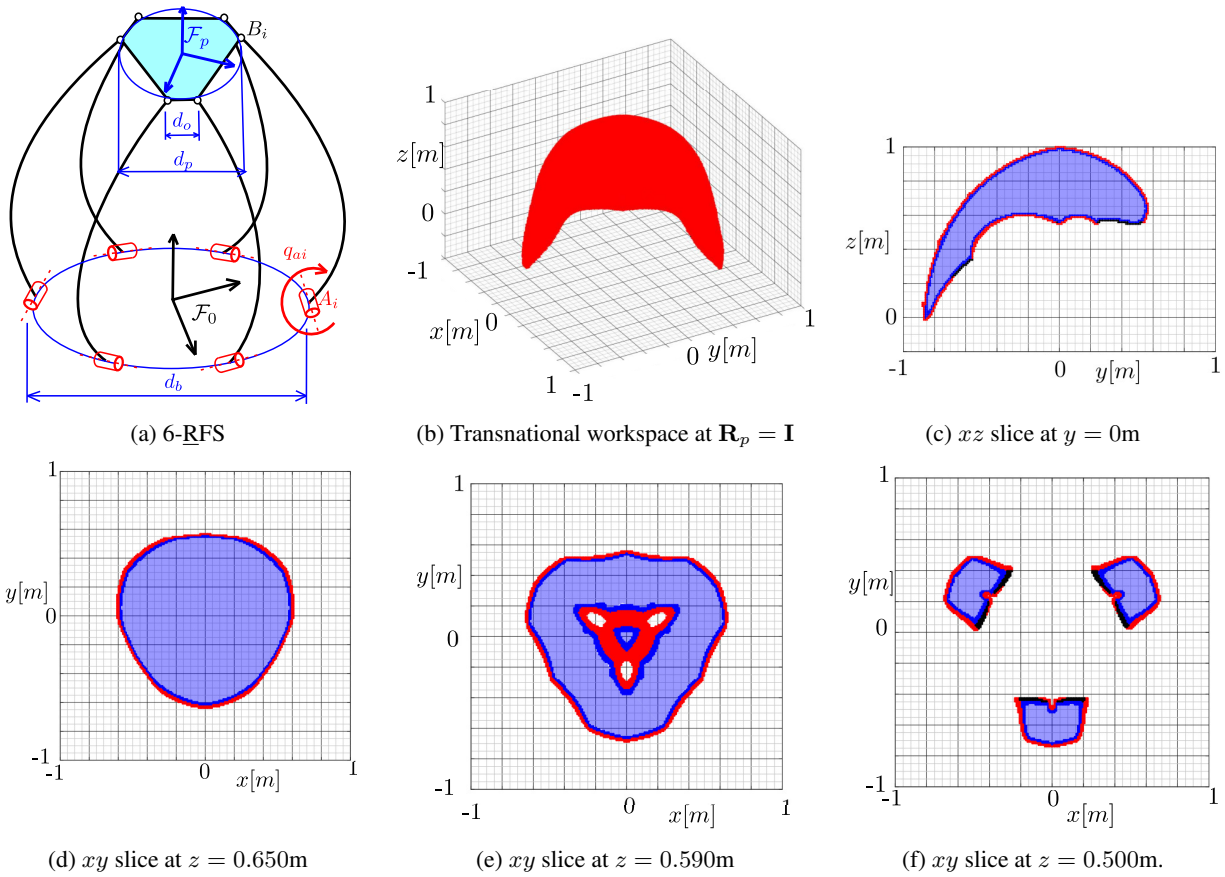
(e) Slice at $\theta = -90°$

Fig. 9: Workspace computation of the $3 - \underline{P}FR$ robot: the robot schematics (a), total orientation workspace in $\theta \in [-\pi, 0]$ (b), constant orientation slices at $\theta = -30°$ (c) ,$-80°$ (d) ,$-90°$ (e). Type-1 singularities are represented in red, results obtained with the approach of [10] in dark blue, and boundaries generated by actuator limits are represented in green.

chosen as 0.4 m. No external forces and gravitational effects are included.

At first, the results obtained by the *BFA* are compared with the flooding algorithm of [10] which computes the full workspace, and not the borders only. The xy plane is discretized with a sampling size $s_g = 5$mm over the range $[-1, +1]$ m in both directions. Then, we initialize τ so that $s_g\tau$ is approximately half of the grid size, that is $\tau = 200$ (in accordance with the heuristic proposed in Sec. 3.2.1). By choosing $n_{exp} = 8$, we obtained the results reported in Fig. 7b, where workspace borders caused by Type-1 and Type-2 singularities are depicted in black, red, the results of [10] are reported in dark-blue, and the exploration trajectories in light blue. The computational time is significantly reduced, passing from approximately 12 mins [10] to 48 s for the *BFA*. Also, a different working mode has been considered by starting from a second initial robot configuration obtained with a different constant-curvature initial guess. The *BFA* has been employed to obtain the workspace depicted in Fig. 7c, the obtained results are in accordance with the one provided by [10] (dark-blue in Fig. 7c), and the computational time has been reduced from 130 s to 31 s. We also investigated the influence of the material strain limit on the workspace size, as shown in Fig. 8a: we computed the $\underline{R}FRF\underline{R}$ robot workspace with strain limit equal to 0.60%, 0.75%, 0.90% and the results are superimposed. By increasing the material resistance, the workspace area increases consequently, and regions closer to the workspace center become more accessible.

Then, the *BFA* has been compared with the optimization algorithm proposed in [30]. We decided to compare our approach with [30] because of the interesting performances of the optimization algorithm, its simplicity, and the possible application to *CPRs*. The optimization algorithm is an iterative algorithm for the computation of *CRs* boundaries, based on the selection of some points $\mathbf{v}^*$ placed in regions assumed out of the workspace (Fig. 8b). Then, an optimization problem is set up to find the robot configuration $\mathbf{y}$ for which the distance between the *EE* position $\mathbf{p}_p$ and $\mathbf{v}^*$ is minimum, subjected to the verification of equilibrium Eq. (8). Additional constraints may be included in the optimization problem as well (e.g equilibrium stability, strain limits). The solution of the constrained optimization problem is solved repeatedly with various $\mathbf{v}^*$ to reconstruct the



(a) 6-RFS

(b) Transnational workspace at $\mathbf{R}_p = \mathbf{I}$

(c) xz slice at $y = 0$m

(d) xy slice at $z = 0.650$m

(e) xy slice at $z = 0.590$m

(f) xy slice at $z = 0.500$m.

Fig. 10: Workspace computation of the $6 - \underline{R}FS$ robot: the robot schematics (a), the translational workspace (b), xz slice at $y = 0$m (c), xy slices at $z = 0.65$m (d), $z = 0.59$m (e), $z = 0.50$m (f). Results obtained with the approach of [10] in dark blue, Type-1 and Type-2 singularities are represented in red, and black, respectively.

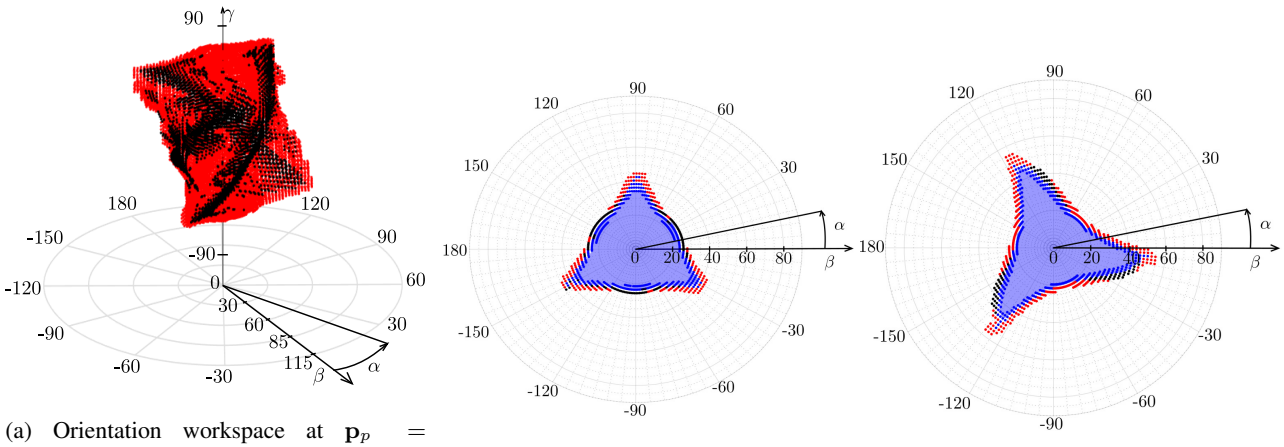
workspace boundary. At first, we focus on the reconstruction of the external workspace border of Fig. 7b: to achieve comparable accuracy in the workspace prediction, the optimization approach required 300 points placed over a circumference of radius of 1.3 m and a total computational time of 22 s whereas our algorithm required 48 s. The optimization approach performs a reduced computational time since it requires the solution of a single optimization problem to find points on the border while the border reconstruction strategy of the *BFA* requires multiple *IGSP* solutions to approximate the border as shown in Fig. 3c. However, the optimization algorithm [30] is not capable to identify holes in the workspace, (as the one present in the $\underline{R}FRF\underline{R}$ robot workspace), and a different placement of points $\mathbf{v}^*$ is required to attempt the inner hole identification. This is a known limitation of the optimization algorithm, previously mentioned in [30] and, in this direction, the *BFA* performs better since it is able to identify the workspace hole by running a second exploration routine.

A second issue worthy to be mentioned is related to the prediction of the exterior border: the optimization approach may fail when the robot admits multiple working modes (Fig. 8c). In general, the optimization approach finds the robot configuration that minimizes the distance between the *EE* and a user-selected outer point, but there is no constraint that prevents the change in the robot working mode. An example of this issue is reported in Fig. 8c: the optimization approach individuates stable solutions that present a shorter distance from points $\mathbf{v}^*$, but these configurations are reachable only by crossing the Type-1 singularity that the optimization approach is not able to individuate. On the other hand, the *BFA* does not suffer this issue, since it is based on the exploration of the boundary by successive iteration in the vicinity of a previously detected boundary.

4.2 3-PFR robot

This subsection investigates the workspace of a $3 - \underline{P}FR$ robot. In this case study the capability of investigating workspace that involves the position and the orientation simultaneously. Following the terminology of [9], we seek to compute the total orientation workspace of a planar *CPR*.

The $3 - \underline{P}FR$ robot has been the focus of the authors previous work [22] where its workspace has been analyzed: in this work, a different arrangement of the prismatic actuators has been employed (Fig. 9a) with the goal of obtaining wider workspaces. The $3 - \underline{P}FR$ robot has three prismatic actuators ($\underline{P}$): we denote with $L_r = 2$ m the finite length of the actuators. The beam extremity is actuated by the movement of the rails and the beam tip is passively connected to



(a) Orientation workspace at $\mathbf{p}_p = [0; 0; 0.8]m$

(b) Zero torsion workspace.

(c) Constant torsion slice at $\gamma = 25°$

Fig. 11: Workspace computation of the $6 - \underline{R}FS$ robot: the orientation workspace at $\mathbf{p}_p = [0; 0; 0.8]$ (a), the zero-torsion workspace (b), the constant torsion workspace at $\gamma = 25°$ (c). Results obtained with the approach of [10] in dark blue, Type-1 and Type-2 singularities are represented in red, and black, respectively.

a platform of diameter $d_P = 0.15$ m through passive revolute joints. Joint limits are taken into account during the workspace evaluation.

In order to investigate the orientation ability of the $3 - \underline{P}FR$ robot over the workspace, we evaluate the total orientation workspace by exploring the end-effector position $\mathbf{p} = [x, y]$ and orientation θ over a three-dimensional grid that discretizes $xy\theta$. The xy plane is sampled with a grid size of 10 mm, while θ is discretized by a 2° sampling over the range $[-180, 0]°$, and $\tau = 100$, $n_{exp} = 20$ according to the heuristics of Sec. 3.2.1. The resulting total orientation workspace is displayed in Fig. 9b. Even if the workspace is connected in the interval $\theta \in [-90, 0]°$, by increasing the *EE* orientation, the constant orientation workspace splits into three non-connected components, as highlighted in Figs. 9d,9e: these slices may be difficult to compute with state-of-the art approaches based on the iterative computation of several xy slices with different orientations. Concerning the computational time comparison, the *BFA* required 53 mins. while 4h 33 min are required with the approach of [10].

4.3 6-RFS robot

This subsection investigates the workspace of a $6-\underline{R}FS$ robot. As commonly done in parallel robots [9], there is limited interest in the evaluation of the 6-*DoF* capabilities simultaneously and frequently position and orientation abilities are evaluated separately. Thus, following the terminology of [9], this case study shows the possibility of efficiently evaluating translational workspace and orientation workspace of spatial *CPRs*.

The $6-\underline{R}FS$ robot (Fig. 10a) was previously characterized in [10], and it has six revolute actuators that actuate the beams proximal section placed at the robot base over a circle of diameters $d_B = 0.8$ m. Flexible beams are connected to a rigid platform through passive spherical joints (S), such as the one employed in the design of [56]. The platform diameter is $d_P = 0.4$ m, and its overall mass is 100 g. Beams are arranged over the platform as described in Fig. 10a, with $d_o = 0.1$ m being the distance of adjacent joints on the same corner. Platform weight and legs gravity aligned along $-z$ are considered during the simulations.

The investigation of the $6 - \underline{R}FS$ robot workspace starts with the translational workspace computation where the platform orientation is fixed as $\mathbf{R}_p = \mathbf{I}_3$. A three-dimensional uniform grid samples the xyz space with grid limits of $[-1, +1]$ m at each direction and a sampling size of 10 mm. The exploration parameters are selected as $\tau = 100$ and $n_{exp} = 20$. The translational workspace is displayed in Fig. 10b: by employing the *BFA* we reduced the computational time to 2h and 30 mins in comparison to the 14 h 30 mins required by [10]. Moreover, state-of-the-art approaches based on slice evaluation of the workspace boundaries may difficultly evaluate the robot workspace in cases of slices as the one represented in Fig.10e,10f since holes or not connected components are present.

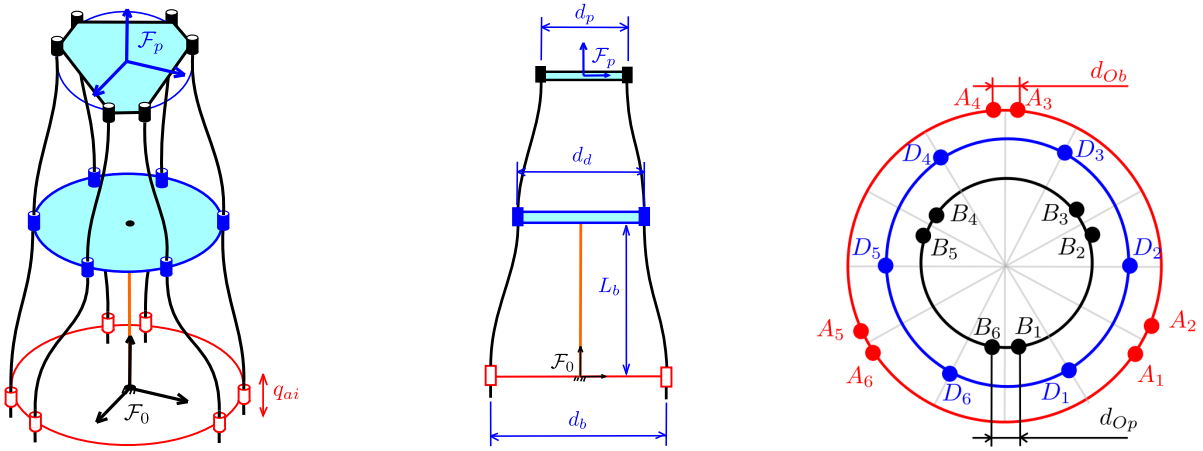
Then, the orientation workspace of the $6-\underline{R}FS$ robot is evaluated at $\mathbf{p}_P = [0, 0, 0.8]$ m. To perform the evaluation, a Tilt-and-Torsion orientation parametrization is employed [57], and the platform rotation matrix $\mathbf{R}_p$ is defined as:

$$\mathbf{R}_p = \mathbf{R}_a(\alpha, \beta)\mathbf{R}_z(\gamma) \tag{23}$$

$$\mathbf{R}_a = \mathbf{R}_z(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(-\alpha) \tag{24}$$

with $\mathbf{R}_z, \mathbf{R}_y$ being elementary rotation matrices and α, β, γ three orientation angles: while α, β defines the tilt of the platform, the angle γ defines it torsion. A uniform grid of step 2° discretized the angle values, with $\alpha \in [-180, 180]°, \beta \in [0, 90]°, \gamma \in [-90, 90]°$. In this case, to define τ, we considered the longer task-space direction (α direction), which results in a higher τ and more complex exploration paths. Thus, $\tau = 45$ and $n_{exp} = 20$.

The resulting orientation workspace, displayed in cylindrical coordinates, is reported in Fig. 11a. Concerning the



(a) $6-\underline{F}R$ with an intermediate constaint. (b) Side view. (c) Top view.

Fig. 12: The $6-\underline{F}R$ with an intermediate constraint. The robot architecture is shown in (a); side and top views of the robot are proposed in (b), and (c) respectively, to illustrate beam connections and relevant design dimensions.

computational time, the *BFA* required 1 h and 52 mins while the flooding algorithm [10] employed 10 h and 3 mins. Then, the zero Torsion slice is extracted from the volume and reported in Fig. 11b. Maximum tilt abilities ($\beta = 59°$) are reached with $\gamma = 25°$ (Fig. 11c).

4.4 6-FR with an intermediate constraint

This subsection focuses on the workspace evaluation of a *CPR* with a more complex structure than the previous case studies to further demonstrate the generality of the *BFA*. We focus on a $6-\underline{F}R$ with an intermediate constraint (Fig. 12a), similar to the prototype of [32]. As before, we investigate position and orientation abilities separately.

The $6-\underline{F}R$ has six linear actuators that vary the lengths of the beams placed at the robot base over a circle of diameters $d_B = 0.12$ m (Fig. 12b). Beam distal sections are connected to a rigid platform through passive revolute joints (R), with platform diameter $d_P = 0.08$ m, and an overall mass of 100 g. An intermediate disk of diameter $d_d = 0.10$ m, which prevents large nonlinear beam deflections [32], is placed between the base and the platform. Cylindrical pairs are mounted over the disk and, differently from [32], the disk is mounted over a passive flexible beam of constant length L_b. Beams are arranged as described in Fig. 12c: the i-th beam is actuated by the linear motor installed on A_i. Then, the beam passes through the cylindrical disk pair placed in D_i, and the distal section of the beam is connected to the platform revolute joint in B_i. The distances between adjacent joints on the same corner are $d_{Op} = d_{Ob} = 0.02$ m for the platform and the base, respectively. Platform and disk weights, aligned along $-z$, are considered during the simulations. We consider linear actuator bounds of $[0.2, 1.0]$ m.

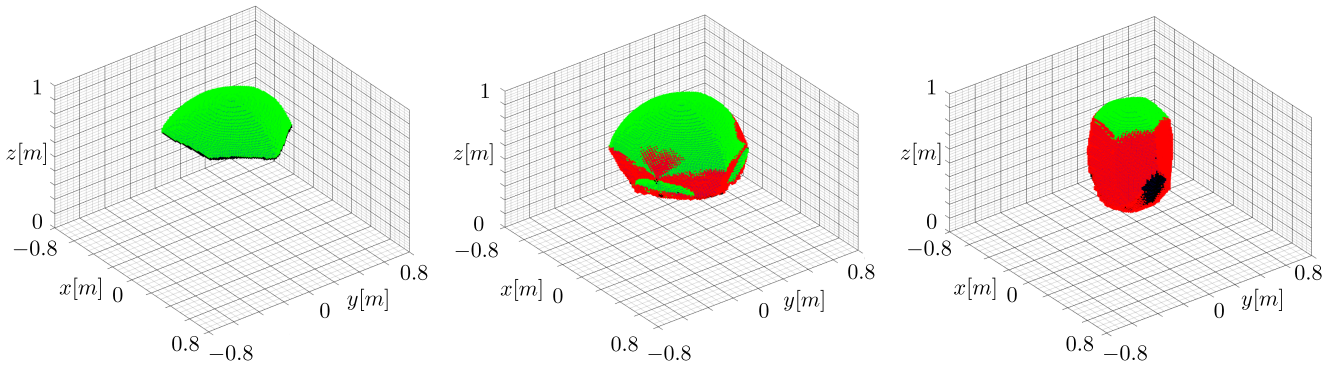
As done for the previous case study, we investigate position and orientation capabilities separately. We focus on the translational workspace computation, with platform orientation fixed as $\mathbf{R}_p = \mathbf{I}$. A three-dimensional uniform grid of dimension $[-1, +1]$ in each direction samples the xyz space, with a sampling size of 10 mm in each direction and τ, n_{exp} initialized to $100, 20$, respectively.

We computed the translational workspace by considering three different values of L_b Fig. 13) and, for each simulation, we considered the workspace volume (obtained by using the *boundary* Matlab function) to measure the workspace extension. With L_b =0.6 m (Fig. 13a), the *BFA* required approximately 1h 40 mins compared to the approximately 12 h of [10]. The workspace volume, mainly delimited by actuators limits boundary, is 0.0137 m^3. Instead, a larger volume of 0.0376 m^3 is obtained by lowering the disk to L_b =0.4 m (Fig. 13b). In this case, the *BFA* required 4h 46 mins and the flooding algorithm of [10] almost 28 h. By further reducing L_b to 0.2 m, we obtain the workspace represented in Fig. 13c. The workspace volume is reduced to 0.0175 m^3 and the required computational time is 202 min for the *BFA* (almost 13h for [10]). In both L_b =0.2, L_b =0.6 we obtained similar workspace volumes, but the computational time drastically differs: for $L_b = 0.2$, the workspace is mainly delimited by Type-1 singularities where the solver requires several iterations to converge while, for $L_b = 0.6$, the boundary is defined mostly by mechanical limits where the *IGSP* can be solved with reduced computational cost.

Concerning the orientation capabilities, we describe the platform orientation by employing a Tilt-and-torsion description and the platform orientation matrix $\mathbf{R}_p$ is obtained by Eq. (24). Figure 14 compares zero torsion workspace at $\mathbf{p}_p = [0; 0; 0.8]$ by varying L_b. The maximum tilt angle of $\beta = 48°$ is reached with L_b =0.6 m (Fig. 14a), but the orientation abilities are more uniform w.r.t. α if L_b =0.4 m (Fig. 14b). The orientation abilities are instead reduced for L_b =0.2 m (Fig. 14c).

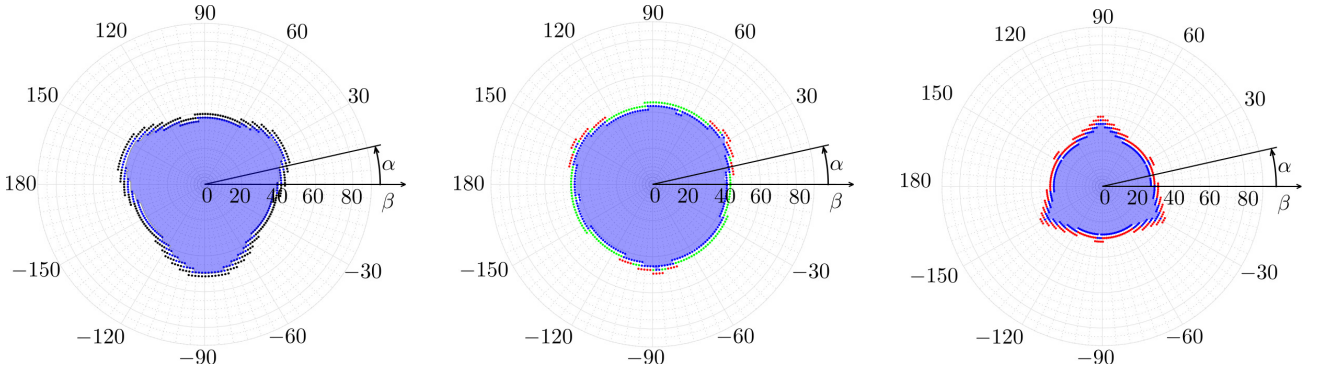
5 Conclusion

In this paper, we proposed an algorithm for the computation of workspace boundaries of *CPRs*. Our algorithm, based



(a) $L_b = 0.6$ m (b) $L_b = 0.4$ m (c) $L_b = 0.2$ m

Fig. 13: Comparison of translational workspaces at $\mathbf{R}_p = \mathbf{I}$ by varying L_b. Type-1 singularities are represented in red, Type-2 singularities in black, results obtained with the approach of [10] in dark blue, and boundaries generated by actuator limits are represented in green.



(a) $L_b = 0.6$ m (b) $L_b = 0.4$ m (c) $L_b = 0.2$ m

Fig. 14: Comparison of zero torsion workspaces at $\mathbf{p}_p = [0; 0; 0.8]$m by varying L_b. Type-1 singularities are represented in red, Type-2 singularities in black, results obtained with the approach of [10] in dark blue, and boundaries generated by actuator limits are represented in green.

on a free-space exploration strategy and on a boundary reconstruction algorithm, reduced the computational time w.r.t. to actuation or task-space discretization strategies by identifying only the boundaries of *CPRs*' workspace. Additionally, the *BFA* included several kinds of constraints such as singularities, equilibrium stability, joint and material limits, all simultaneously during the workspace computation. In comparison to state-of-the-art boundary computation approaches, the *BFA* provided a useful tool to identify holes in the workspace that may occur in *CPRs*: this is possible thanks to the proposed space exploration strategy. With the aim of being a general workspace evaluation tool, the *BFA* works with *CPRs* modelling strategies based on general discretization assumptions, which increases the algorithm generality. Four case studies demonstrated the effectiveness of the proposed approach in terms of i) computational-time reduction w.r.t. actuation or task-space discretization approach, ii) general applicability and results correctness that some state-of-the-art approach may not achieve. We believe our algorithm is well suited for design explorations, where the reduced computational time and the algorithm generality are relevant advantages. In this scenario, the user explores the robot workspace various times by changing the values of some design variables. The user stops the process when the robot workspace volume and shape satisfy some task-driven requirements.

6 Limitations and future development.

In this Section, we stress the main limitations of our work. First, we recognize that the use of attractive points is effective, but it may fail when parameters are defined improperly. As any heuristic methods, we acknowledge that parameter tuning is critical for the best performance of the algorithm. For our algorithm, the number of explorations n_{exp} and the exploration parameter τ require trial-and-error tuning. At the end of Section 3, we discussed how to possibly tune the parameters. However, we do not exclude that better heuristics may be defined.

Even though our algorithm is able to identify internal workspace boundaries (namely, holes in the workspace) thanks to the proposed exploration strategy, there is no certainty of identifying all of them. However, to the authors knowledge, no state-of-the-art algorithm for the boundary

computation of continuum robots demonstrated the ability to identify holes in the workspace. A correct tuning of τ, n_{exp} gives higher assurance to find all the workspace borders, but no proof is available.

The computational time of our methods is higher than the one of highly specialized boundary computation algorithms (e.g. [30]), usually defined for a single class of *CRs*, or even a single *CR*. In the authors opinion, this limitation is the price to be paid for a more generally applicable algorithm, with the additional benefits outlined in the paper.

Future work will be directed toward the application of the *BFA* for more complex workspace explorations. We believe that a possible extension of the *BFA* for the reachable workspace computation may be envisioned. For example, a first exploration of the translational workspace (orientation parameters are fixed to ϕ_1) may be conducted to find a point on the workspace boundary. Then, at the boundary location, the end-effector orientation is explored by using the boundary flooding algorithm for the orientation workspace exploration. If there exists a set of the orientation parameters ϕ_2 where the robot lies within the workspace, ϕ_2 may be selected as the new orientation parameters, and the translational boundaries further explored.

References

[1] Burgner-Kahrs, J., Rucker, D. C., and Choset, H., 2015. "Continuum robots for medical applications: A survey". *IEEE Transactions on Robotics,* **31**(6), pp. 1261–1280.

[2] Kolachalama, S., and Lakshmanan, S., 2020. "Continuum robots for manipulation applications: A survey". *Journal of Robotics,* **1**, pp. 1–19.

[3] Bryson, C. E., and Rucker, D. C., 2014. "Toward parallel continuum manipulators". In 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 778–785.

[4] Gilbert, H. B., Hendrick, R. J., and Webster III, R. J., 2015. "Elastic stability of concentric tube robots: A stability measure and design test". *IEEE Transactions on Robotics,* **32**(1), pp. 20–35.

[5] Zhong, Y., Hu, L., and Xu, Y., 2020. "Recent advances in design and actuation of continuum robots for medical applications". *Actuators,* **9**(4), p. 142.

[6] Wu, G., and Shi, G., 2019. "Experimental statics calibration of a multi-constraint parallel continuum robot". *Mechanism and Machine Theory,* **136**, pp. 72–85.

[7] Coyle, S., Majidi, C., LeDuc, P., and Hsia, K. J., 2018. "Bio-inspired soft robotics: Material selection, actuation, and design". *Extreme Mechanics Letters,* **22**, pp. 51–59.

[8] Rao, P., Peyron, Q., Lilge, S., and Burgner-Kahrs, J., 2021. "How to model tendon-driven continuum robots and benchmark modelling performance". *Frontiers in Robotics and AI,* **7**, p. 223.

[9] Merlet, J.-P., 2005. *Parallel robots*, Vol. 128. Springer Science & Business Media.

[10] Briot, S., and Goldsztejn, A., 2022. "Singularity conditions for continuum parallel robots". *IEEE Transactions on Robotics,* **38**(1), pp. 507–525.

[11] Till, J., and Rucker, D. C., 2017. "Elastic stability of cosserat rods and parallel continuum robots". *IEEE Transactions on Robotics,* **33**(3), pp. 718–733.

[12] Amehri, W., Zheng, G., and Kruszewski, A., 2020. "FEM based workspace estimation for soft robots: a forward-backward interval analysis approach". In 2020 3rd IEEE International Conference on Soft Robotics (RoboSoft), New Haven, CT, USA, pp. 170–175.

[13] Trivedi, D., Lesutis, D., and Rahn, C. D., 2010. "Dexterity and workspace analysis of two soft robotic manipulators". In Proceedings of the ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 2: 34th Annual Mechanisms and Robotics Conference, Parts A and B. Montreal, Quebec, Canada, pp. 1389–1398.

[14] Burgner-Kahrs, J., Gilbert, H. B., Granna, J., Swaney, P. J., and Webster, R. J., 2014. "Workspace characterization for concentric tube continuum robots". In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, pp. 1269–1275.

[15] Orekhov, A. L., Black, C. B., Till, J., Chung, S., and Rucker, D. C., 2016. "Analysis and validation of a teleoperated surgical parallel continuum manipulator". *IEEE Robotics and Automation Letters,* **1**(2), pp. 828–835.

[16] Singh, I., Singh, M., Pathak, P. M., and Merzouki, R., 2017. "Optimal work space of parallel continuum manipulator consisting of compact bionic handling arms". In 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), Macau, Macao, pp. 258–263.

[17] Nuelle, K., Sterneck, T., Lilge, S., Xiong, D., Burgner-Kahrs, J., and Ortmaier, T., 2020. "Modeling, calibration, and evaluation of a tendon-actuated planar parallel continuum robot". *IEEE Robotics and Automation Letters,* **5**(4), pp. 5811–5818.

[18] Altuzarra, O., Caballero, D., Campa, F. J., and Pinto, C., 2019. "Position analysis in planar parallel continuum mechanisms". *Mechanism and Machine Theory,* **132**, pp. 13–29.

[19] Mauze, B., Dahmouche, R., Laurent, G. J., Andre, A. N., Rougeot, P., Sandoz, P., and Clevy, C., 2020. "Nanometer precision with a planar parallel continuum robot". *IEEE Robotics and Automation Letters,* **5**(3), pp. 3806–3813.

[20] Pan, H., Chen, G., Kang, Y., and Wang, H., 2021. "Design and kinematic analysis of a flexible-link parallel mechanism with a spatially quasi-translational end effector". *Journal of Mechanisms and Robotics,* **13**(1), p. 011022.

[21] Lilge, S., Nuelle, K., Boettcher, G., Spindeldreier, S., and Burgner-Kahrs, J., 2020. "Tendon actuated continuous structures in planar parallel robots: A kinematic analysis". *Journal of Mechanisms and Robotics,* **13**(1),

p. 011025.

[22] Zaccaria, F., Ida, E., Briot, S., and Carricato, M., 2022. "Workspace computation of planar continuum parallel robots". *IEEE Robotics and Automation Letters,* **7**(2), pp. 2700–2707.

[23] Chablat, D., and Wenger, P., 1998. "Working modes and aspects in fully parallel manipulators". In Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146), Leuven, Belgium, Vol. 3, IEEE, pp. 1964–1969.

[24] Wu, G., and Shi, G., 2022. "Design, modeling, and workspace analysis of an extensible rod-driven parallel continuum robot". *Mechanism and Machine Theory,* **172**, p. 104798.

[25] Abdel-Malek, K., and Yeh, H.-J., 1997. "Analytical boundary of the workspace for general 3-*DoF* mechanisms". *The International Journal of Robotics Research,* **16**(2), pp. 198–213.

[26] Wang, Y., Wu, Z., Wang, L., Feng, B., and Xu, K., 2022. "Inverse kinematics and dexterous workspace formulation for 2-segment continuum robots with inextensible segments". *IEEE Robotics and Automation Letters,* **7**(1), pp. 510–517.

[27] Haug, E. J., Luh, C.-M., Adkins, F. A., and Wang, J.-Y., 1996. "Numerical Algorithms for Mapping Boundaries of Manipulator Workspaces". *Journal of Mechanical Design,* **118**(2), pp. 228–234.

[28] Amehri, W., Zheng, G., and Kruszewski, A., 2021. "Workspace boundary estimation for soft manipulators using a continuation approach". *IEEE Robotics and Automation Letters,* **6**(4), pp. 7169–7176.

[29] Snyman, J., Du Plessis, L., and Duffy, J., 2000. "An optimization approach to the determination of the boundaries of manipulator workspaces". *J. Mech. Des.,* **122**(4), pp. 447–456.

[30] Amehri, W., Zheng, G., Kruszewski, A., and Renda, F., 2021. "Discrete cosserat method for soft manipulators workspace estimation: An optimization-based approach". *Journal of Mechanisms and Robotics,* **14**(1).

[31] Amehri, W., Zheng, G., and Kruszewski, A., 2022. "Fem-based exterior workspace boundary estimation for soft robots via optimization". *IEEE Robotics and Automation Letters*.

[32] Orekhov, A. L., Aloi, V. A., and Rucker, D. C., 2017. "Modeling parallel continuum robots with general intermediate constraints". In 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, IEEE, pp. 6142–6149.

[33] Zaccaria, F., Briot, S., Chikhaoui, M. T., Idà, E., and Carricato, M., 2021. "An analytical formulation for the geometrico-static problem of continuum planar parallel robots". In *ROMANSY 23 - Robot Design, Dynamics and Control CISM. ROMANSY 2020. International Centre for Mechanical Sciences*, G. Venture, J. Solis, Y. Takeda, and A. Konno, eds., Vol. 601. Springer International Publishing, Cham, pp. 512–520.

[34] Antman, S., 1995. *Nonlinear Problems of Elasticity*, Vol. 107. Springer Verlag New York.

[35] Ziegler, H., 2013. *Principles of structural stability*, Vol. 35. Birkhäuser.

[36] Meier, C., Popp, A., and Wall, W. A., 2014. "An objective 3d large deformation finite element formulation for geometrically exact curved kirchhoff rods". *Computer Methods in Applied Mechanics and Engineering,* **278**, pp. 445–478.

[37] Webster, R. J., and Jones, B. A., 2010. "Design and kinematic modeling of constant curvature continuum robots: A review". *The International Journal of Robotics Research,* **29**(13).

[38] Gravagne, I. A., Rahn, C. D., and Walker, I. D., 2003. "Large deflection dynamics and control for planar continuum robots". *IEEE/ASME transactions on mechatronics,* **8**(2), pp. 299–307.

[39] Jones, B. A., and Walker, I. D., 2006. "Kinematics for multisection continuum robots". *IEEE Transactions on Robotics,* **22**(1), pp. 43–55.

[40] Renda, F., Cacucciolo, V., Dias, J., and Seneviratne, L., 2016. "Discrete cosserat approach for soft robot dynamics: A new piece-wise constant strain model with torsion and shears". In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea (South), IEEE, pp. 5495–5502.

[41] Renda, F., Boyer, F., Dias, J., and Seneviratne, L., 2018. "Discrete cosserat approach for multisection soft manipulator dynamics". *IEEE Transactions on Robotics,* **34**(6), pp. 1518–1533.

[42] Renda, F., and Seneviratne, L., 2018. "A geometric and unified approach for modeling soft-rigid multibody systems with lumped and distributed degrees of freedom". In 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, IEEE, pp. 1567–1574.

[43] Boyer, F., Lebastard, V., Candelier, F., and Renda, F., 2021. "Dynamics of continuum and soft robots: A strain parameterization based approach". *IEEE Transactions on Robotics,* **37**(3), pp. 847–863.

[44] Peyron, Q., Rabenorosoa, K., Andreff, N., and Renaud, P., 2018. "Evaluation of dynamic relaxation to solve kinematics of concentric tube robots". In *Advances in Robot Kinematics 2018. ARK 2018. Springer Proceedings in Advanced Robotics*, J. Lenarcic and V. Parenti-Castelli, eds., Vol. 8. Springer International Publishing, pp. 100–107.

[45] Peyron, Q., Boehler, Q., Rabenorosoa, K., Nelson, B. J., Renaud, P., and Andreff, N., 2018. "Kinematic analysis of magnetic continuum robots using continuation method and bifurcation analysis". *IEEE Robotics and Automation Letters,* **3**(4), pp. 3646–3653.

[46] Duriez, C., 2015. "Control of elastic soft robots based on real-time finite element method". In 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, IEEE, pp. 3982–3987.

[47] Romero, I., 2008. "A comparison of finite elements for nonlinear beams: the absolute nodal coordinate and geometrically exact formulations". *Multibody System Dynamics,* **20**(1), pp. 51–68.

[48] Simo, J., and Vu-Quoc, L., 1986. "A three-dimensional finite-strain rod model. Part II: Computational aspects". *Computer methods in applied mechanics and engineering,* **58**(1), pp. 79–116.

[49] Sadati, S. M. H., Naghibi, S. E., Walker, I. D., Althoefer, K., and Nanayakkara, T., 2018. "Control space reduction and real-time accurate modeling of continuum manipulators using ritz and ritz–galerkin methods". *IEEE Robotics and Automation Letters,* **3**(1), pp. 328–335.

[50] Schillinger, D., Evans, J. A., Reali, A., Scott, M. A., and Hughes, T. J., 2013. "Isogeometric collocation: Cost comparison with galerkin methods and extension to adaptive hierarchical nurbs discretizations". *Computer Methods in Applied Mechanics and Engineering,* **267**, pp. 170–232.

[51] Orekhov, A. L., and Simaan, N., 2020. "Solving cosserat rod models via collocation and the magnus expansion". In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, IEEE, pp. 8653–8660.

[52] Briot, S., and Boyer, F., 2022. "A geometrically exact assumed strain modes approach for the geometrico- and kinemato-static modelings of continuum parallel robots". *IEEE Transactions on Robotics*, p. doi:10.1109/TRO.2022.3219777.

[53] Renda, F., Armanini, C., Lebastard, V., Candelier, F., and Boyer, F., 2020. "A geometric variable-strain approach for static modeling of soft manipulators with tendon and fluidic actuation". *IEEE Robotics and Automation Letters,* **5**(3), pp. 4006–4013.

[54] Boyer, F., Lebastard, V., Candelier, F., and Renda, F., 2022. "Extended hamilton's principle applied to geometrically exact kirchhoff sliding rods". *Journal of Sound and Vibration,* **516**, p. 116511.

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

[56] Böttcher, G., Lilge, S., and Burgner-Kahrs, J., 2021. "Design of a reconfigurable parallel continuum robot with tendon-actuated kinematic chains". *IEEE Robotics and Automation Letters,* **6**(2), pp. 1272–1279.

[57] Bonev, I. A., and Ryu, J., 2001. "A new approach to orientation workspace analysis of 6-*DoF* parallel manipulators". *Mechanism and machine theory,* **36**(1), pp. 15–28.