Implicit Active Constraints for Safe and Effective Guidance of Unstable Concentric Tube Robots

Konrad Leibrandt, Christos Bergeles, and Guang-Zhong Yang, *Fellow, IEEE*

***Abstract*— Safe and effective telemanipulation of concentric tube robots is hindered by their complex, non-intuitive kinematics. In order for clinicians to operate these robots naturally, guidance schemes in the form of attractive and repulsive constraints can simplify task execution. The real-time seamless calculation and application of guidance, however, requires computationally efficient algorithms that solve the non-linear inverse kinematics of the robot and guarantee that the commanded robot configuration is stable and sufficiently away from the anatomy. This paper presents a multi-processor framework that allows on-the-fly calculation of optimal safe paths based on rapid workspace and roadmap precomputation. The real-time nature of the developed software enables complex guidance constraints to be implemented with minimal computational overhead. A clinically challenging user study demonstrates that the incorporated guiding constraints are highly beneficial for fast and accurate navigation with concentric tube robots.**

I. Introduction

Continuum robots have unique advantages when used in surgical procedures that require navigation through delicate anatomical cavities or convoluted pathways [1]. Concentric Tube Robots (CTRs) are a representative of continuum robots whose shape is controlled by rotating and telescopically translating pre-curved super-elastic tubes [2]. Their size is comparable to that of catheters, but are able to apply distant forces and exhibit shape control. Therefore, they are used in similar clinically-critical interventions, such as navigating the vasculature to reach the heart [3], and snaking through the nasal cavity [4], the unary tracts [5], or the bronchia [6].

CTR mechanics have thus far been investigated extensively, with reliable models existing for torsionally compliant kinematics and equilibrium conformations [7], [8]. The latter result from the instability of CTR, which under certain configurations rapidly release energy after sufficient torsional wind-up of their tubular components [9]. These instabilities cause uncontrollable tube motion, which, if it occurs intraoperatively, may cause severe tissue trauma. For this reason, computational design, path-planners, and inverse kinematics solvers consider criteria not only to avoid such configurations but also to guide the human operator away from them [10], [11], [12], [13].

All aforementioned approaches rely on fast solvers of the inverse kinematics, so that the reachable workspace and path-plans may be evaluated in real-time. Common approaches to achieve real-time performance for inverse kinematics include root-finding and look-up tables [7], and jacobian-based approximations [14], sparse path-plans using rapidly-exploring-random trees for pre-defined targets [15], and pre-calculation of dense-path plans for the robot workspace [16], [17]. The latter contribution allows interactive rate motion planning, but does not consider robot instability. The computational overhead of pre-calculating the inverse kinematics in a dense workspace while remaining along stable collision-free paths is vast, and, so far, has required tens-of-hours of computation. Complexity increases when instability comes into consideration.

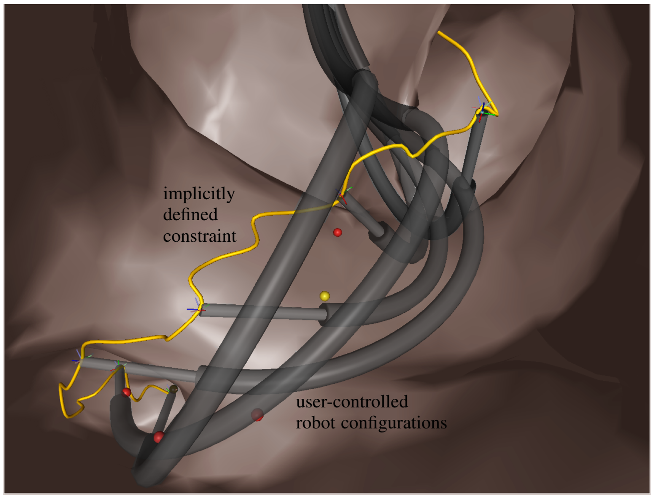

Fig. 1. Concentric tube robots guided on implicitly defined active constraints. Using collaborative control deep seated targets are reached safe and efficiently.

In addition to assistance via path planning, CTR are particularly appealing for guidance under active constraints (AC) / virtual fixtures (VF) since they are deployed in sensitive enviroments and are not intuitive to operate [18]. AC/VF allow semi-automatic robot telemanipulation, wherein the complexity of the inverse kinematics is hidden to the user while he/ she is guided to avoid collisions with the anatomy or unstable configurations. AC comprise three components: a) constraint generation, b) constraint evaluation, and c) constraint enforcement [19].

Implicit active constraints (IAC) contribute towards constraint generation [20]. While in conventional AC a human explicitely defines the constraint, e.g. by segmenting pre-operative images and considering the resulting polygon meshes as obstacles, or by learning from demonstrations of

Konrad Leibrandt was supported by the Imperial College PhD Scholarship Scheme and EPSRC UK. Konrad Leibrandt and Guang-Zhong Yang are with the Hamlyn Centre for Robotic Surgery, Imperial College London, London, United Kingdom. Christos Bergeles is with the Translational Imaging Group, Centre for Medical Image Computing, University College London, London, United Kingdom. Corresponding author: `k.leibrandt12@imperial.ac.uk`

common trajectories, the benefit of IACs is that they algorithmically create task-relevant constraints based on limited user input by leveraging information about the defined surgical tasks [21].

This work contributes to IAC for CTR by proposing constraint generation based on safe on-line path planning for collaborative control. Our algorithms build on [12]. We leverage distributed computing to provide globally dense workspace roadmaps for CTRs in minutes, and combine on-demand path-planning with on-line local inverse-kinematics to guide the user along safe trajectories. A user study simulating a challenging clinical scenario from neurosurgery compares our current and previous algorithms, and demonstrates improved performance in computational complexity, total robot-tip-path length, and procedure time.

II. Kinematic Model, Anatomical and Stability Constraints

Manipulation of CTR involves translating and rotating sections of super-elastic pre-curved tubes. Independently translating robot sections can comprise either one or two tubes, forming constant, or variable curvature sections [7]. The translation of tube i is denoted as ${}_{i}\varphi$, and its rotation at the base as ${}_{i}\alpha^{\mathrm{B}}$. A variable-curvature section has three degree-of-freedom (DoF) (${}_{1}\varphi$=${}_{2}\varphi$, ${}_{1}\alpha^{\mathrm{B}}$, ${}_{2}\alpha^{\mathrm{B}}$), a constant curvature section has two DoF (${}_{1}\varphi$, ${}_{1}\alpha^{\mathrm{B}}$). The set of $\left\{{}_{i}\varphi,{}_{i}\,\alpha^{\mathrm{B}}\right\}$ for $i \in [1, N_{\mathrm{t}}]$, where N_{t} is the number of tubes, comprise the joint space $\boldsymbol{q} = \left[\boldsymbol{\varphi}, \boldsymbol{\alpha}^{\mathbf{B}}\right]$. CTR models based on torsionally compliant kinematics, *i.e.* tube twisting, have no closed form solutions for the forward or inverse kinematics. The underlying model we are refering to in this work assume the absence of external load.

The forward kinematics represents a boundary value problem, which is solved by calculating the solutions from a torsion-free tip angle ${}_{i}\alpha^{\mathrm{T}}$ to the base angles ${}_{i}\alpha^{\mathrm{B}}$ and then using the set of joint values $\boldsymbol{q} = \left[\boldsymbol{\varphi}, \boldsymbol{\alpha}^{\mathbf{B}}\right]$, matrix exponentials and calculated curvatures along the robot curve estimate its shape. To solve the forward kinematics, a discretization ϵ_{Arc} of the robot spine is necessary. Thorough explanations of CTR kinematics can be found in [7], [2].

A. Anatomical Constraints

Anatomical constraints are derived from a multi-polygonal surface representing tissue or organs. Each polygon is subdivided until the distance between all its defining vertices is below ϵ_{Lat}. The resulting vertex set represents the anatomy Γ and is stored in a k-d tree (3-d tree) to allow fast proximity queries.

To calculate the distance from the robot to anatomy Γ the generated 3-d tree is queried to determine the nearest neighbour to each point on the robot centreline ${}_{i_{\mathrm{cl}}}\boldsymbol{d}^{\mathrm{NN}}$. The collision distance between the robot and anatomy d_{col} is then estimated as

$$d_{\mathrm{col}} = \min_{i_{\mathrm{cl}}} \left({}_{i_{\mathrm{cl}}}\boldsymbol{d}^{\mathrm{NN}} - \left\| \frac{1}{2} [\epsilon_{\mathrm{Arc}}, \epsilon_{\mathrm{Lat}}]^{T} \right\|_{2} - {}_{i_{\mathrm{cl}}}r^{t} \right), \tag{1}$$

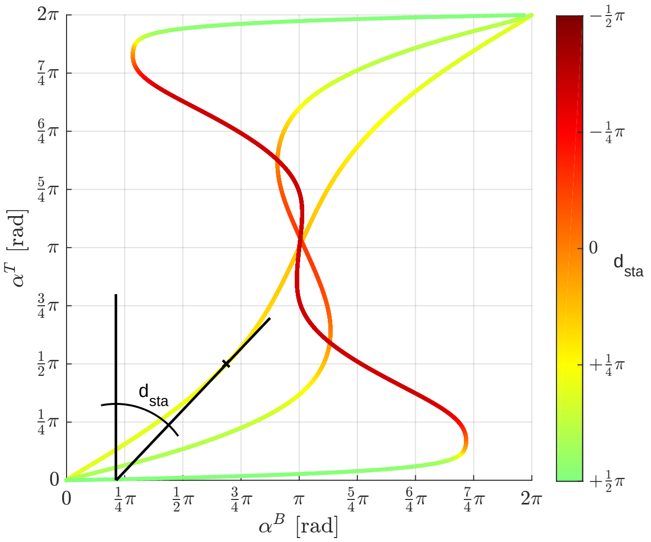

Fig. 2. S-surve of $\alpha^{B} \rightarrow \alpha^{T}$. The tangent slope on the S-curve is a measure for the stability. The color of the curve is based on the quantification of stability (4).

where ${}_{i_{\mathrm{cl}}}r^{t}$ is the tube radius at the respective centre line point. Negative values of d_{col} mark configurations that collide with the anatomy.

B. Stability Constraints

Stability, in a binary fashion, is calculated online according to [10], [12]. A quantative stability measure for a given tube i is here calculated as

$$ {}_{i}\sigma({}_{i}\alpha^{B}) = \begin{cases} \min\limits_{\substack{\forall_{i}\alpha^{B'} \\ \in [0,{}_{i}\alpha^{B}]}} \left(\left(\left. \frac{\partial\, {}_{i}\alpha^{T}}{\partial\, {}_{i}\alpha^{B}} \right|_{{}_{i}\alpha^{B} = {}_{i}\alpha^{B'}} \right)^{-1} \right), \text{ if } {}_{i}\alpha^{T} \leq \pi \\ \min\limits_{\substack{\forall_{i}\alpha^{B'} \\ \in [{}_{i}\alpha^{B},2\,\pi]}} \left(\left(\left. \frac{\partial\, {}_{i}\alpha^{T}}{\partial\, {}_{i}\alpha^{B}} \right|_{{}_{i}\alpha^{B} = {}_{i}\alpha^{B'}} \right)^{-1} \right), \text{ if } {}_{i}\alpha^{T} > \pi \end{cases} \tag{2}$$

and the stability for a given robot configuration is found by examining all interacting-tube combinations

$$\sigma^{q}\left(\mathbf{q}\right) = \min_{i \in [1, N_{\mathrm{t}}]} \left(\sigma_{i}\right). \tag{3}$$

A robot configuration is stable if $\sigma^{q} \geq 0$. In (2) the inverse of the tangent slope along the S-curve relating base angle α^{B} and tip angle α^{T} is minimized, because a CTR configuration is unstable if the slope at its operation point on the S-curve is negative. When a CTR is approaching an instability, the slope (first derivative) of the S-curve approaches $+\infty$. Further, in an unstable configuration the slope increases from $-\infty$. This discontinuity does not perform well when arising in the optimisation problems tackled here. Using the inverse of the slope, solves this issue. The distance from an unstable configuration can be quantified based on the angle of the slope

$$d_{\mathrm{sta}} = \frac{\pi}{2} - \mathrm{atan2}\left(1, \sigma^{q}\right). \tag{4}$$

The distance measure d_sta represents the signed angle distance of the most critical slope ($\measuredangle = \operatorname{atan2}(1, \sigma^q)$), i.e. after examining the S-curves of all subsequent-tube combinations, to the vertical tangent ($\measuredangle = \frac{\pi}{2}$). Fig. 2 shows three exemplary S-curves. In line with the collision distant measurement, negative values of d_sta represent unstable configurations. This quantification of proximity to instability allows the definition of differentiable cost-functions considering collision and instabilities, which is crucial for the generation of implicit constraints.

III. Implicit Constraints

Implicit constraints are algorithmically generated based on knowledge of the task and limited user input. Ideally, the user's intention is sufficient to define a constraint [20]. In the following, a probabilistic roadmap approach is used to generate on-line constraints based on the user simply specifying the desired surgical target.

A. Constraint Generation - Probabilistic Roadmap

In the following the construction of an undirected graph $G^{\Gamma,\mathcal{T}}$, with vertices $v_i \in G^{\Gamma,\mathcal{T}}$ representing safe CTR configurations and edges $e_{i,j}$ connecting v_i with v_j is presented. The edges of the graph represent the possibility of transitioning from one configuration to the other. Querying this graph in real-time will provide path-plans that generate the IAC.

First, random sampling is used to find safe CTR configurations that constitute the graph's vertices. Second, local planning is performed to connect those vertices with edges. These processes are performed offline, but, contrary to existing works, complete in minutes due to the exploitation of parallel computer architectures. After the creation of the graph, online querying is used to find shortest paths between the current robot configuration and the surgical targets.

1) Sampling: Workspace sampling is performed by many computing nodes, which are controlled by a master node. The master node sends to each slave node (the node carrying out the actual computations) the kinematic definition of the CTR $\mathcal{T}$, the anatomical constraints (Γ), and parameters to control robot configuration sampling.

The joint-space ($\boldsymbol{q}$) is sampled using the mersenne twister pseudorandom number generator (MT19937-32). Its state sequence is fully seeded using the entropy pool of the computer operating system. The uniformly distributed pseudo-random numbers $q_\mathrm{r,u} \in [0, 1]$ are scaled to translational and joint-values (φ) using

$$q_\mathrm{r} = {}_i\varphi^{\max} \, {q_\mathrm{r,u}}^{\gamma_\varphi}, \; \gamma_\varphi \in [0, 1], \tag{5}$$

where ${}_i\varphi^{\max}$ is the curved length of a robot-section, and rotational joint-values

$$q_\mathrm{r} = 2\pi \left(q_\mathrm{r,u} - {}^1\!/_2\right) \tag{6}$$

respectively. The γ^φ-scaling for the translational joints is introduced because uniform joint-space sampling does not create a uniformly sampled task-space, see Fig. 3. Since the sweep area of the end-effector increases as the robot is extended, there need to be more samples for elongated CTRs. Furthermore, extended robots are more likely to collide with the anatomy, and hence those samples are more likely to get rejected, further reducing their density. Dense sampling for elongated robot configurations is thus achieved through γ^φ, which promotes the investigation of elongated robot configurations. Although an optimal sampling approach may exist, the proposed heuristic performs sufficiently well in practice.

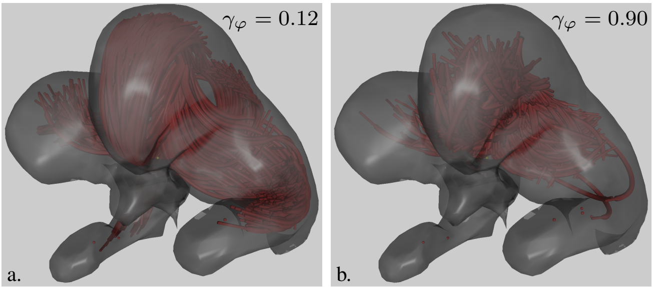

Fig. 3. Random concentric tube configurations, 1000 configurations each. a. Scaling: $\gamma_\varphi = 0.12$, b. Scaling: $\gamma_\varphi = 0.90$.

The robot shape is calculated for each random joint-value sample $\boldsymbol{q}_r$. This examined robot-configuration is accepted if the distance to the anatomy (1) and the distance to instabilities (4) is below a threshold

$$d_\mathrm{col}\left(\boldsymbol{q}_r\right) \le d_\mathrm{col}^\mathrm{thres} \wedge d_\mathrm{sta}\left(\boldsymbol{q}_r\right) \le d_\mathrm{sta}^\mathrm{thres}. \tag{7}$$

Accepted configurations are transmitted to the master node, where ther are added as a vertices to $G^{\Gamma,\mathcal{T}}$.

The three sampling-control parameters $\mathcal{T}^\varphi$, $d_\mathrm{sta}^\mathrm{thres}$, $d_\mathrm{sta}^\mathrm{thres}$ are controlled by the master node and periodically multicasted to all available slave nodes.

B. Local Planar

Once a desired/ sufficient number of vertices is added to the undirected graph $G^{\mathcal{T},\Gamma}$, a parallel local planar is initiated to connect the vertices in $G^{\mathcal{T},\Gamma}$ with edges. To align with graph-theory terminology, we will use the terms «vertex» and «robot configuration» interchangeably. Further, «vertex position» refers to the end-effector position of the robot configuration; «vertex joint-values» refers to the joint-values of the robot configuration; and «vertex centre-line» refers to the centre-line of the robot configuration.

Let:

- $f_\mathrm{ee}(v_j, v_k)$ be a scalar function providing the euclidean norm between two vertex positions v_j and v_k;
- $\boldsymbol{g}_\mathrm{jv}(v_j, v_k)$ be a vector function providing the joint-value differences of two vertices;
- $g_\mathrm{jv}^\mathrm{thres}(v_j, v_k, \Delta_\alpha, \Delta_\varphi) \in \{0, 1\}$ a function giving 1 if all rotational joints differ less than Δ_α, while translational joints differ less than Δ_φ;
- $h_\mathrm{cl}(v_j, v_k)$ the root-mean-square (rms) error of the euclidean distance of two vertex centre-lines.

In order to penalise centre-lines of different lengths, the last function is evaluated by iterating over both centre-lines simultaneously while summing the squared euclidean distances along all discretisation points. If a center-line contains fewer points, its end-effector position is used for the remaining iterations.

An edge $e_{j,k}$ between v_j and v_k is inserted if:

$$\begin{aligned}&\big(f_{\text{ee}}(v_j,v_k)\geq\epsilon^{min}\big)\wedge\big(f_{\text{ee}}(v_j,v_k)\leq\epsilon^{max}\big)\\ &\wedge\big(g_{\text{jv}}^{\text{thres}}(v_j,v_k,\Delta_\alpha,\Delta_\varphi)>0\big)\\ &\wedge\big(h_{\text{cl}}(v_j,v_k)<\epsilon_{\text{cl}}^{\text{thres}}\big).\end{aligned}\tag{8}$$

The motivation for these restrictions is: a) the end-effectors positions of connected vertices/ configurations should be close; b) the differences in joint-space should be limited, ensuring small joint-velocities when transitioning from one configuration to the next one; and c) vertex centre-line should be similar, as encoded in (8). Note that these hard restrictions to edge insertion guarantee a sparse graph, which results in faster shortest-path queries. The edge weight $(\omega\left(e_{j,k}\right))$ is

$$\begin{aligned}&\omega\left(v_j,v_k\right)=\omega\left(e_{j,k}\right)\\ &=f_{\text{ee}}(v_j,v_k)+\boldsymbol{\omega}_{jv}^{T}\,\boldsymbol{g}_{\text{jv}}(v_j,v_k)+\omega_{cl}\,h_{\text{cl}}(v_j,v_k)\end{aligned}\tag{9}$$

where $\boldsymbol{\omega}_{jv}$ is a weighting vector for the joint-value differences, and ω_{cl} a weight for the root-mean-square error of the centre-line differences.

Considering a vertex position v_j, the goal is to obtain edges connecting to vertexes in different directions, which will encode the possibility to transition from one robot configuration to another in heterogeneous task space directions. Consequently, neighbouring vertices v_k of v_j are associated with one octant of the coordinate-system centered at v_j. The edge $e_{j,k}$ is associated with the same octant as v_k. Those edges with the k_E-lowest edge-costs per octant are included into the graph representation $G^{\mathcal{T},\Gamma}$.

C. Querying of Roadmap - IAC

To convert online an implicit describtion into an explicit constraint definition, source and goal vertices need to be set to trigger a shortest-path search on $G^{\mathcal{T},\Gamma}$. The source vertex $v^{\text{s}}\in G^{\mathcal{T},\Gamma}$ is found by applying for the current robot configuration v^{op} and for all vertices in the graph $(\forall v_i\in G^{\mathcal{T},\Gamma})$ the criteria (8) and subsequently choosing as source vertex the one which minimizes the cost (9) to v^{op}. The target position (in 3d-task-space) to which active guidance should be planned to is $\boldsymbol{p}^T$. Within $d^{T,\text{thres}}$ from $\boldsymbol{p}^T$, there are many vertices since the considered CTRs are redundant and the sampling is dense. The set of those vertices is $\mathcal{V}_{\text{can}}^{T}$, and contains robot configurations with an end-effector position closer to $\boldsymbol{p}^T$ than the threshold $d^{T,\text{thres}}$.

Ideally, the shortest paths leading from the source vertex v^{s} to all of the vertices within the set $\mathcal{V}_{\text{can}}^{T}$ should be examined, and the best of all the shortest paths (each corresponding to an examined target-configuration) should be selected. In practice, this strategy would result in severe computational overhead, stemming from the fact that most vertices within the set would have configurations that are very similar and thus offer limited information-gain when examined. Assuming that an increased difference between two robot configurations implies a difference in the arising shortest paths, the chosen way to find the optimal path is to iteratively examine the configuration in $\mathcal{V}_{\text{can}}^{T}$ that is most distant to the set of already examined target vertices $\mathcal{V}^T$. In other words, the set $\mathcal{V}^T$ is initialised with the vertex closest to $\boldsymbol{p}^T$. Then, iteratively, the vertex v_k most distant to the set of vertices $\mathcal{V}^T$ according to (9) is marked as additional target-vertex, i.e.

$$v_k\in\mathcal{V}_{\text{can}}^{T}\;\Big|\;v_k\text{ maximize}\sum_{\forall v_j\in\mathcal{V}^T}\omega\left(v_k,v_j\right)\tag{10}$$

is added iteratively to $\mathcal{V}^T$ until its cardinality reaches an upper limit $N_s^{\max}$ or $\left|\mathcal{V}^T\right|=\left|\mathcal{V}_{\text{can}}^{T}\right|$.

The shortest path from the source vertex to each target vertex in $\mathcal{V}^T$ is independently and in parallel calculated using the A^*-algorithm, where the admissible A^*-heuristic function is the euclidean distance between two given vertex positions. The heuristic can be weighted to increase search speed at the expense of the optimality of the shortest path. Note, the computational time to determine source and target vertices is marginal.

The end-effector positions along the best of the shortest path is used as the basis of the guidance constraint. In a last step, an interpolation between all returned vertex positions is performed using centripetal catmull-rom splines to generate a C^2 function as smooth guidance constraint.

D. Constraint Evaluation and Enforcement

Guidance and forbidden-region constraints are evaluated at every time point. Guidance forces are only rendered when the robot is not colliding with the anatomy, and when either approaching surgical targets or when motion along a path-plan is demanded.

To guide the user along the shortest path between two robot configurations, the distance between user-input $\boldsymbol{p}^D$ and the path plan has to be calculated. An exhaustive search calculating the shortest distance of the line-segments connecting the discrete points of the interpolated shortest path and $\boldsymbol{p}^D$ is used. The shortest distant vector is used to enforce the guidance constraint.

To provide the user with haptic feedback when colliding with the anatomy, the shortest distance between $\boldsymbol{p}^D$ and the anatomy is calculated by querying the 3-d tree that represents the anatomy Γ. The shortest vector is used to generate a repulsive force, similar to [12].

Constraint enforcement is implemented by rendering forces to the haptic device used to telemanipulate the robot. An elasto-plastic friction model is used to generate the forces [18], [22], [12], due to its advantages of avoiding oscillations, instabilities, and redirecting kinetic energy to enforce constraints. Please refer to [12], [18] for a thorough explanation, omitted here due to lack of space. The parameterization of the algorithm is depicted in Table I.

IV. On-Line Inverse Kinematics

When the user manipulates the robot, the path plans only engage in form of guiding constraints. The inverse kinematics are constantly solved, in real-time, to guide the user towards the path and estimate collisions with the anatomy and potential instabilities.

TABLE I

FRICTION CONSTRAINT PARAMETERS

Parameter (FRVF)		**Value**	**Parameter (GVF)**		**Value**
f_C	[N]	3.5	f_C	[N]	1.75
Θ_{appex}	[°]	30	Θ_{appex}	[°]	15
σ_0	[N/mm]	1.0	σ_0	[N/mm]	0.5
σ_1	[N.s/mm]	0.0006	σ_1	[N.s/mm]	0.0003
σ_2	[N.s/mm]	0.0024	σ_2	[N.s/mm]	0.0012
z_{css}	[mm]	3.5	z_{css}	[mm]	3.5

TABLE II

ROBOT PARAMETERS USED FOR NAVIGATION IN VENTRICLES

Robot Section Type	**Curvature [mm^{-1}]**	**Length [mm]**	**Stiffness Ratio**
Variable Curvature	0.0522	66.47	5:1
Fixed Curvature	0.05263	66.70	1:1
Zero Curvature	0.0	12.65	0.05:1

Inverse kinematics are calculated using a solver improved over our previously presented work in [12]. A set of different optimization algorithms is used in parallel to minimize the cost,

$$\begin{aligned} \mathcal{C}\left(\boldsymbol{q},\mathcal{T},\Gamma,\boldsymbol{p}^D,\boldsymbol{z}^D\right) &= \gamma_{\text{pos}}\ \left|\boldsymbol{p}^D-\boldsymbol{p}^O\right|_2 \\ &+ \gamma_{\text{ori}}\ \operatorname{asin}\left(\left|\boldsymbol{z}^D-\boldsymbol{z}^O\right|_2\right) \\ &+ \gamma_{\text{col}}\ \mathcal{C}_{\text{col}}\left(\boldsymbol{q},\mathcal{T},\Gamma\right) \\ &+ \gamma_{\text{sta}}\ \mathcal{C}_{\text{sta}}\left(\boldsymbol{q},\mathcal{T}\right) \end{aligned} \tag{11}$$

where $\mathcal{C}_{\text{col}}$ is the collision cost and $\mathcal{C}_{\text{sta}}$ the stability cost. They are defined as,

$$\mathcal{C}_{\text{col}}\left(\boldsymbol{q},\mathcal{T},\Gamma\right)=\begin{cases} 1 & \text{, if } d_{\text{col}} \leq d_{\text{col}}^{\text{low}} \\ \frac{d_{\text{col}}-d_{\text{col}}^{\text{up}}}{d_{\text{col}}^{\text{low}}-d_{\text{col}}^{\text{up}}} & \text{, if } d_{\text{col}}^{\text{low}} < d_{\text{col}} \leq d_{\text{col}}^{\text{up}} \\ 0 & \text{, if } d_{\text{col}}^{\text{up}} < d_{\text{col}} \end{cases} \tag{12}$$

and

$$\mathcal{C}_{\text{sta}}\left(\boldsymbol{q},\mathcal{T}\right)=\begin{cases} 1 & \text{, if } d_{\text{sta}} \leq d_{\text{sta}}^{\text{low}} \\ \frac{d_{\text{sta}}-d_{\text{sta}}^{\text{up}}}{d_{\text{sta}}^{\text{low}}-d_{\text{sta}}^{\text{up}}} & \text{, if } d_{\text{sta}}^{\text{low}} < d_{\text{sta}} \leq d_{\text{sta}}^{\text{up}} \\ 0 & \text{, if } d_{\text{sta}}^{\text{up}} < d_{\text{sta}} \end{cases} \tag{13}$$

where γ_{pos}, γ_{ori}, γ_{col}, and γ_{sta} are cost-weights determining position and orientation accuracy in relation to collision and stability costs. The parameters $d_{\text{col}}^{\text{up}}$, $d_{\text{col}}^{\text{low}}$, $d_{\text{sta}}^{\text{up}}$, and $d_{\text{sta}}^{\text{low}}$ determine how collision and stability costs increase from a lower bound to an upper bound. Each optimizer i_{opt} reports the optimal joint-values ${}_{i_{\text{opt}}}\boldsymbol{q}$ for the desired end-effector pose $[(\boldsymbol{p}^D)^T,(\boldsymbol{z}^D)^T]^T$. The central computing node is comparing the operating cost,

$$\boldsymbol{q}_{\text{opt}}=\begin{cases} {}_{i_{\text{opt}}}\boldsymbol{q}, \text{if } \mathcal{C}\left({}_{i_{\text{opt}}}\boldsymbol{q},\mathcal{T},\Gamma,\boldsymbol{p}^D,\boldsymbol{z}^D\right) \\ \qquad < \mathcal{C}\left(\boldsymbol{q}_{\text{opt}},\mathcal{T},\Gamma,\boldsymbol{p}^D,\boldsymbol{z}^D\right) \\ \boldsymbol{q}_{\text{opt}}, \text{else} \end{cases} \tag{14}$$

whenever receiving a new optimal joint-value from a computing node. Since the desired pose changes continuously the cost for the operating optimal cost, $\boldsymbol{q}_{\text{opt}}$, has to be constantly recalculated. This approach allows different cycle times for different optimizers, which is usefull since global non-linear optimizers require in general longer than local.

A. Inverse Kinematics without Path-Plan Guidance

When the user does not demand for a path-plan-based guidance the initial joint-values $\boldsymbol{q}_{\text{ini}}$ used for each inverse-kinematics optimizer are the operational joint-values $\boldsymbol{q}_{\text{opt}}$. Those initial-joint values are constantly updated and multi-casted by the central node to all optimizer nodes.

B. Inverse Kinematics with Guidance Constraints

In case there is demand for guidance constraints, the inverse-kinematics optimizers are provided with initial joint-configurations based on the path-plan. The initial joint-values $\boldsymbol{q}_{\text{ini}}$ of the vertex closest to the current end-effector position are multi-casted to the optimizer nodes, which results in directing the optimizers towards joint-values ${}_{i_{\text{opt}}}\boldsymbol{q}$ close to those robot-configurations provided by the shortest path-plan. Not taking the path-plan configurations directly ensures that the user is always in full control while assuming that the user's intention is to follow the path-plan.

V. IMPLEMENTATIONAL DETAILS AND EXPERIMENT

In the following section implementation details and experimental results are outlined to demonstrate the effectiveness of the approach.

A. Kinematic Model

The forward-kinematics of the torsionally compliant model was implemented in C++ using `template`-meta programming techniques, vector-instructions for costly trigonometric functions, optimizing conditional expressions leading to branching, and ensuring cache-coherency. The concentric tube robot-object is continously stored in memory, which allows fast transmission over the network and simple storage of large numbers of the different configurations comprising $G^{\mathcal{T},\Gamma}$. No compromises were made regarding precision or run-time usability.

A comparison is performed against our previous C++ implementation used in [12], CTR parameters are depicted in Table II. The upper-part of the statistic in Table III was created with the Cachegrind-software package whereas as the same pseudo-random joint-value samples for the current and previous implementation were used. The overall speed-up of the current implementation to the previous one is 1,411.0% in a single-thread measurement. It shows that a well designed memory structure improved the

TABLE III

FORWARD KINEMATICS IMPLEMENTATION IMPROVEMENTS

Rel. profiler results:	**L1-cache**	**LL-cache**	**D1-cache**	**Branching**
current/previous	10.9%	87.2%	14.4%	5.9%
Robot configurations:	**1,000**	**10,000**	**100,000**	**1,000,000**
$\epsilon_{\text{Arc}} =$ 1 mm	16 ms	160 ms	1,632 ms	16,331 ms
$\epsilon_{\text{Arc}} =$ 0.1 mm	167 ms	1600 ms	16,323 ms	163,527 ms

TABLE IV
PROBABILISTIC SAMPLING RESULTS
TIME FOR SUCCESFULL SAMPLES [S] VS. ACCEPTANCE RATE [%]

Translation scaling (γ_φ)	**cluster I.** **250,000**	**cluster I.** **1,000,000**	**cluster II.** **250,000**	**cluster II.** **1,000,000**
0.1	1003.4 s 0.045 %	3792.8 s 0.045 %	107.9 s 0.044 %	550.8 s 0.046 %
0.4	24.6 s 2.52 %	73.3 s 2.53 %	21.5 s 2.38 %	81.1 s 2.55 %
0.7	9.4 s 10.9 %	39.3 s 10.9 %	16.9 s 10.6 %	70.4 s 10.7 %
1.0 (uniform)	9.0 s 21.0 %	33.8 s 21.1 %	17.4 s 20.9 %	60.7 s 20.4 %
Parametrization:	$\epsilon_{\text{Arc,FK}}$ 1 mm	$\epsilon_{\text{Arc,Sta}}$ 2 mm	$\epsilon_{\alpha,Sta}$ 1 °	$d_{\text{col}}^{\text{thres}}$ 1 mm	$d_{\text{sta}}^{\text{thres}}$ 5 °

cache coherency and that the `template`-meta programming techniques reduced the amount of conditional expressions, which reduced costly code-execution-branching drastically.

The second part of Table III, was generated using the implemenation single-threaded on an Intel® Xeon® CPU E5-2637 v3. It shows that the implemenation scales linearly with increased sample size and also linearly to the reciprocal of the descretization step-size ϵ_{Arc}.

Table III demonstrates that carefully tuned implementations can lead to massive computation improvements which in turn enables new algorithmic approaches.

B. Probabilistic Sampling

Distributed sampling was performed with two different sets of computing hardware. Cluster I consists of 4 Intel® Xeon® CPU E5-2680 v2 with each 2 Intel® Xeon Phi™ Coprocessor 7120P. Cluster II fascilitates 210-230 8-core office-machines via the HTCondor framework. The number of machines used in cluster II fluctuated during the benchmarking, which is normal due to how the HTCondor framework works. In both cases sampling was performed as described in Sec, III-A.1. For inter-node communication the Spread toolkit, a high-performance messaging service resilient to faults across local and wide area networks, was used. Different types of faults (failures of single nodes or entire node segments) in the communication were simulated to ensure the resilence of the developed framework.

In Table IV timings for creating valid configuration samples for the roadmap are listed, including parametrization regarding discretization of the forward-kinematics and the stability check, as well as parameters for the configuration acceptance criteria in (7).

C. User Study - Navigation in Hydrocephalic Ventricles

The clinical procedure descibed in [12], [10] is used in our case study due to the complex anatomy and elongated and curved CTR. In summary, the CTR end-effector has to reach the lateral horns of the brain's ventricles and cauterise a continuous path between the two. Cauterisation reduces the production of cerebrospinal fluid (CSF) and limits hydrocephalus pressure (see Fig. I) thus reducing the inflation of the ventricles..

Manouvering unstable CTRs safely to difficult to reach caveties, increases operational time compared to when stability is ignored [12]. Further, without path-planning, users need to explore different ways to reach into the horns of the ventricles, which can be frustrating and time consuming. Therefore, the performed study evaluates the benefits of on-demand IAC against inverse-kinematics without guidance.

In our experimental study, 14 participants (4 female) were asked to maneuver the robot through the ventricles and cauterize 13 targets. Participants were allowed to use the path-planning on-demand during the «active constraints mode». During the «free-mode» participants were not allowed to use guidance. The start mode was altered between participants to eliminate learning effects of the task.

a) Graph Generation: A graph with 1,048,576 vertices was generated for the user-study, with 209,715 vertices per translation scaling $\gamma_\varphi \in [0.12, 0.25, 0.35, 0.70, 0.90]$. The time required to acquire all valid robot configurations amounted to 149.4 s (using both clusters combined). The local planning according to Sec. III-B took 37.4 s (single-node, 20 cpu) to populate the roadmap graph with 8,337,665 edges. The maximum branching factor was chosen to $k_E = 2$, which allows 16 edges per vertex (2 towards each octant). The overall pre-computation time for the global graph in the experiment sums up to roughly 3 min, requiring approximately 1.7 GB of memory. This is a drastic improvement compared to the several hours required by the current state-of-the-art, with less dense road-maps . Further parameters of the probabilistic roadmap generation are provided in Table V.

In addition, three local graphs with 131,072 vertices were calculated to cover individual parts of the workspace. Local graphs were partioned based on vertex distances to the robot base. They could be queried independently and in parallel to the global graph for the shortest path.

b) On-Line Inverse Kinematics: The on-line inverse kinematic concept of [12] was employed. Overall, 56 differently parameterized non-linear optimizers distributed over 5 compute nodes were used. The optimization algorithms used are listed in Table VI. Optimization time was limited to 20 ms for all algorithms. The same discretization parameters $\epsilon_{\text{Arc,FK}} = 0.3\,\text{mm}$, $\epsilon_{\text{Arc,Sta}} = 1\,\text{mm}$, $\epsilon_{\alpha,Sta} = 2\,°$ were used for all algorithms. Parameters regarding the cost-function ($\mathcal{C}$,

TABLE V
INVERSE KINEMATIC AND GRAPH GENERATION PARAMETERS

Opti. Cost Eq. (11)		**Value**	**Edge Criteria Eq. (8)**		**Value**
γ_{pos}	[1/mm]	1000.0	ϵ^{min}	[mm]	0.2
γ_{ori}	[1/°]	0.0	ϵ^{max}	[mm]	4.0
γ_{col}	[1]	0.5	Δ_φ	[mm]	5.0
γ_{sta}	[1]	0.5	Δ_α	[°]	40.0

Opti. Cost Eq. (12, 13)		**Value**	**Edge Weight Eq. (9)**		**Value**
$d_{\text{col}}^{\text{low}}$	[mm]	0.5	$\omega_{jv,\varphi}$	[1/m]	15.0
$d_{\text{col}}^{\text{up}}$	[mm]	1.0	$\omega_{jv,\alpha}$	[1/°]	0.0056
$d_{\text{sta}}^{\text{low}}$	[°]	0.0	ω_{cl}	[1/m]	20.0
$d_{\text{sta}}^{\text{up}}$	[°]	5.0			

TABLE VI

OPTIMIZATION ALGORITHMS

Algorithm		**Scope**
BOBYQA	bound optimization by quadratic approximation	local
NEWUOA	new unconstrained optimization algorithm	local
Nelder-Mead	downhill simplex method	local
PRAXIS	principal axis	local
$\mathcal{J}_{\text{damp}}$	damped least-square pseudo-inverse	local
MLSL	multi-level single-linkage	global
CRS2	controlled random search with local mutation	global

(11)) are listed in Table V. The $\mathcal{J}_{\text{damp}}$ algorithm reported intermediate results at the inter-node communication rate of 1 kHz. Each iteration / optimization step of $\mathcal{J}_{\text{damp}}$ took on average 156.3 μs. The accuracy of the on-line kinematics ΔP_{IK} was determined as 0.57 mm, by taking the interquartile-mean of $\Delta P_{\text{ee|sp}} = \|P_{\text{ee}} - P_{\text{sp}}\|_2$ in each time step. The interquartile-mean is more robust towards outliers than the mean. Commanded positions outside of the robot workspace were considered as outliers and did not contribute in the calculation of the error.

c) Result Evaluation: Apart from quantitative evaluations, the user study provided qualitative results on the utility of guidance via a questionarie. The key quantitative metrics were:

i: Duration [min]: Time to complete task.
ii: P_{ee} TD [m]: Travel distance of robot end-effector.
iii: P_{sp} TD [m]: Travel distance of user set position.
iv: $\Delta P_{\text{ee|sp}}$ o.T. [m min]: Time integral over $\|P_{\text{ee}} - P_{\text{sp}}\|_2$.
v: ΔP_{IK} [mm]: Accuracy of the on-line kinematics.
vi: ΔT_{opt} [ms]: Mean time between IK cost improvements.

These metrics are depicted in Fig. 4. The p-values were calculated using the wilcoxon signed-rank test, which shows significant difference in the experiment between the two modes in the metrics i-iv. The results in Fig. 4 indicate an improvement regarding the operation time, and also more efficient maneuvering of the CTR. The distance $\Delta P_{ee|sp}$ for one user is depicted in Fig. 5. The plot shows that using the «free mode» the inverse-kinematic is not able to follow the trajectory specified by the user, resulting in spikes of $\Delta P_{ee|sp}$. This is happening as the user seeks a way of approaching the lateral horns on a trial-and-error basis. Using the enabled «active constraint mode» the distance value is lower and the plot is smoother.

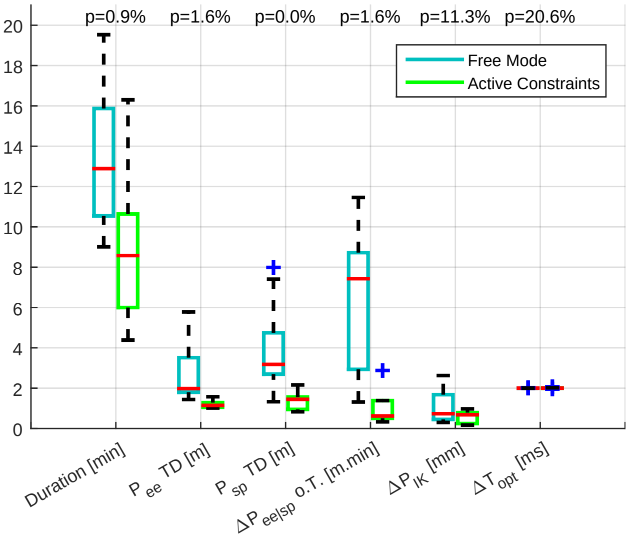

Fig. 4. Metrics of user study. Abscissa: metric type, Ordinate: metric value [depending on type].

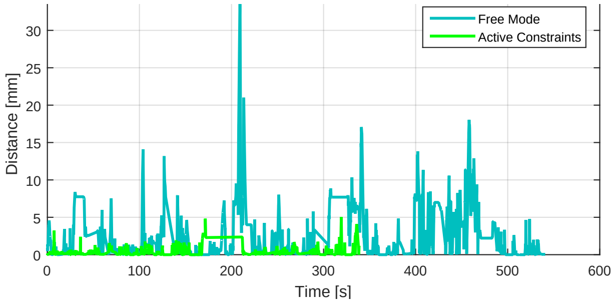

Fig. 5. Distance between robot end-effector P_{ee} and user set-point P_{sp}.

For qualitative evaluation the participants were asked to compare the two modes, on a scale from -3 to $+3$ (-3: negative evaluation, $+3$ positive evaluation). The results are listed in Table VII. All except one user stated that they preferred the «active constraint mode» over the «free mode». The one user preferring the «free mode » performed better in that mode because she found the correct way of approaching the targets at first trial. All other users felt more in-control with guidance enabled, which resulted in a safer feeling. The haptic feedback (guidance and repulsive) were evaluated as useful by all users.

TABLE VII

QUALITATIVE AVERAGE RESULTS OF USER STUDY [−3,+3]

Metric	Free Mode	Active Constraint
Performance effiency	-1.54	1.54
Performance safety	-1.54	2.15
Frustration	-1.46	1.23
Overall performance	-0.69	1.62

General Metric	Evaluation
Guidance Forces	1.85
Forbidden Region Forces	2.15

VI. CONCLUSION

This paper presents implicit active constraints for on-line planning of stable paths to guide a concentric tube robot safely towards arbitrary target positions in a cooperative control approach. A fast method to generate off-line dense path-plans, using distributed computing, was presented. Based on these dense path-plans, shortest-path queries were used to on-line generate active constraint which guide the user. A user study showed significant improvements in the task completion time and the efficients when facilitating path-planning, supporting the value of the developed framework.

The current implementation of the path-plans is limited to static antatomy, which will be addressed in future work.

VII. Acknowledgement

The authors gratefully acknowledge Prof. Pierre Dupont from the Pediatric Cardiac Bioengineering Lab of Boston Children's Hospital, Harvard Medical School for providing the anatomy used in this paper, and the Intel UK Health Innovation team for helping set up the Intel® Xeon Phi$^{\text{TM}}$ cluster.

References

[1] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum robots for medical applications: A survey," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1261–1280, Dec 2015.

[2] R. J. Webster III and B. A. Jones, "Design and kinematic modeling of constant curvature continuum robots: a review," *Int. J. Robotics Research*, vol. 29, no. 13, pp. 1661–1683, 2010.

[3] A. Gosline, N. V. Vasilyev, E. Butler, C. Folk, A. Cohen, R. Chen, N. Lang, P. J. del Nido, and P. E. Dupont, "Percutaneous intracardiac beating-heart surgery using metal MEMS tissue approximation tools," *Int. J. Robotics Research*, vol. 31, no. 9, pp. 1081–1093, 2012.

[4] J. Burgner, D. C. Rucker, H. B. Gilbert, P. J. Swaney, P. T. Russell, K. D. Weaver, and R. J. Webster III, "A telerobotic system for transnasal surgery," *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 3, pp. 996–1006, 2014.

[5] R. J. Hendrick, Richard J and Mitchell, Christopher R and Herrell, S Duke and Webster, "Hand-held transendoscopic robotic manipulators: A transurethral laser prostate surgery case study," *Int. J. Robotics Research*, vol. 34, no. 13, pp. 1559–1572, 2015.

[6] L. G. Torres, R. J. Webster III, and R. Alterovitz, "Task-oriented design of concentric tube robots using mechanics-based models," *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 4449–4455, 2012.

[7] P. E. Dupont, J. Lock, B. Itkowitz, and E. Butler, "Design and control of concentric tube robots," *IEEE Trans. Robotics*, vol. 26, no. 2, pp. 209–225, 2010.

[8] D. C. Rucker, R. J. Webster, G. S. Chirikjian, and N. J. Cowan, "Equilibrium conformations of concentric-tube continuum robots," *The International journal of robotics research*, 2010.

[9] H. B. Gilbert, R. J. Hendrick, and R. J. W. III, "Elastic stability of concentric tube robots: A stability measure and design test," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 20–35, Feb 2016.

[10] C. Bergeles and P. E. Dupont, "Planning stable paths for concentric tube robots," *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 3077–3082, 2013.

[11] C. Bergeles, A. Gosline, N. V. Vasilyev, P. Codd, P. J. del Nido, and P. E. Dupont, "Concentric tube robot design and optimization based on task and anatomical constraints," *IEEE Trans. Robotics*, vol. 31, no. 1, pp. 67–84, 2015.

[12] K. Leibrandt, C. Bergeles, and G.-z. Yang, "On-line collision-free inverse kinematics with frictional active constraints for effective control of unstable concentric tube robots," *IEEE Int. Conf. Robotics and Automation*, pp. 3797–3804, 2015.

[13] R. J. Hendrick, H. B. Gilbert, and R. J. Webster, "Designing snap-free concentric tube robots: A local bifurcation approach," *IEEE Int. Conf. Robotics and Automation*, pp. 2256–2263, 2015.

[14] R. Xu, A. Asadian, A. Naidu, and R. Patel, "Position control of concentric-tube continuum robots using a modified jacobian-based approach," *IEEE Int. Conf. Robotics and Automation*, pp. 5813–5818, 2013.

[15] A. Kuntz, L. G. Torres, R. H. Feins, R. J. Webster, and R. Alterovitz, "Motion planning for a three-stage multilumen transoral lung access system," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, Sept 2015, pp. 3255–3261.

[16] L. G. Torres, C. Baykal, and R. Alterovitz, "Interactive-rate motion planning for concentric tube robots," *IEEE Int. Conf. Robotics and Automation*, pp. 1915–1921, 2014.

[17] L. G. Torres, A. Kuntz, H. B. Gilbert, P. J. Swaney, R. J. Hendrick, R. J. Webster, and R. Alterovitz, "A motion planning approach to automatic obstacle avoidance during concentric tube robot teleoperation," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 2361–2367.

[18] S. Bowyer and F. Rodriguez y Baena, "Dynamic frictional constraints in translation and rotation," *IEEE Int. Conf. Robotics and Automation*, pp. 2685–2692, 2014.

[19] S. A. Bowyer, B. L. Davies, and F. Rodriguez y Baena, "Active constraints/virtual fixtures: A survey," *Robotics, IEEE Transactions on*, vol. 30, no. 1, pp. 138–157, 2014.

[20] K. Leibrandt, H. Marcus, K.-W. Kwok, and G.-Z. Yang, "Implicit active constraints for a compliant surgical manipulator," *IEEE Int. Conf. Robotics and Automation*, pp. 276–283, 2014.

[21] E. Lopez, K.-W. Kwok, C. J. Payne, P. Giataganas, and G.-Z. Yang, "Implicit active constraints for robot-assisted arthroscopy," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5390–5395.

[22] V. Hayward, B. Armstrong, F. Altpeter, and P. Dupont, "Discrete-time elasto-plastic friction estimation," *IEEE Trans. Control Systems*