Implicit Active Constraints for Concentric Tube Robots Based on Analysis of the Safe and Dexterous Workspace

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

Abstract— **The use of concentric tube robots has recognized advantages for accessing target lesions while conforming to certain anatomical constraints. However, their complex kinematics makes their safe telemanipulation in convoluted anatomy a challenging task. Collaborative control schemes, which guide the operator through haptic and visual feedback, can simplify this task and reduce the cognitive burden of the operator. Guaranteeing stable, collision-free robot configurations during manipulation, however, is computationally demanding and, until now, either required long periods of pre-computation time or distributed computing clusters. Furthermore, the operator is often presented with guidance paths which have to be followed approximately. This paper presents a heterogeneous (CPU/GPU) computing approach to enable rapid workspace analysis on a single computer. The method is used in a new navigation scheme that guides the robot operator towards locations of high dexterity or manipulability of the robot. Under this guidance scheme, the user can make informed decisions and maintain full control of the path planning and manipulation processes, with intuitive visual feedback on when the robot's limitations are being reached.**

I. Introduction

Navigating through convoluted pathways to access deep-seated pathologies through natural orifices or single-access ports is a challenging task in minimally invasive surgery (MIS). Continuum robots are ideally suited for these interventions as their shape can be controlled to conform to anatomical constraints [?], [?]. A representative continuum robot is the concentric tube robot (CTR). The manipulation of CTRs encompasses rotation and translation of concentrically arranged, pre-curved super-elastic tubes [?]. The elastic tubes interact with each other creating curved robot shapes and allowing full tip-pose control. The degrees-of-freedom (DoF) of the CTR increase with its number of tubes, making it possible to construct redundant robots wherein the additional DoFs allow for the optimization of the robot shape without compromising tip manipulation.

CTRs have been designed using optimization techniques [?], [?], [?] for several surgical applications such as haemorrhage evacuation [?], tissue approximation for heart surgery [?], bronchoscopic biopsy [?], and transurethral prostatectomy [?].

Analysis of the workspace to describe the reachability of CTRs was presented in [?], while the manipulability of

K. Leibrandt and G.-Z. Yang are with the Hamlyn Centre for Robotic Surgery, Imperial College London, United Kingdom. C. Bergeles is with the Translational Imaging Group, Centre for Medical Image Computing, University College London, United Kingdom. K. Leibrandt was supported by the President's PhD Scholarship of Imperial College London and EPSRC UK. C. Bergeles was supported by an EPSRC-funded UCL Future Leaders Grant. Corresponding author: k.leibrandt12@imperial.ac.uk.

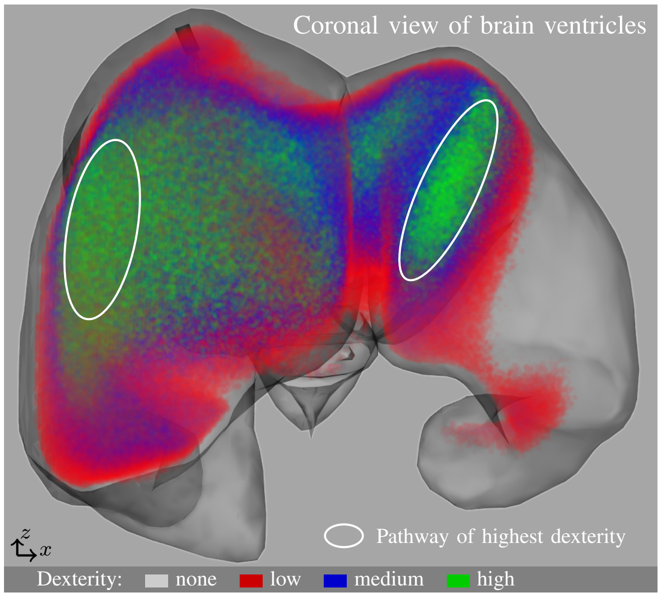
Coronal view of brain ventricles
Pathway of highest dexterity
z
x
Dexterity: none low medium high

Fig. 1. Preoperative analysis of the safe and dexterous workspace of an intraventricular concentric tube robot.

continuum robots was investigated in [?], [?]. Methods to efficiently calculate the Jacobian along the robot body were presented in [?], [?]. The forward and inverse kinematic of CTRs can be solved by accounting for the torsional wind-up along the length of each tube, while enabling computation of metrics describing the stability of configurations. These instabilities arise from torsion itself [?], [?], [?]. Through the torsional wind-up of the tubes, energy accumulates and may rapidly release in fast, uncontrolled changes of the robot shape [?], [?]. Hence, safe manipulation requires the real-time detection of unstable configurations and their evasion by the robot operator [?].

Different methods have been presented to achieve real-time manipulation of CTRs, including look-up tables [?], Jacobian-based approximation [?], sparse path-plans using rapidly-exploring random trees [?], dense path-plans [?], [?], and multi-node parallel kinematics optimizers [?]. First, these approaches either required long pre-computation times, or used inverse kinematics without path-planning and risked non-convergence due to local minima. Second, the existing guidance schemes required the user to approximately follow the provided safe path-plans with limited capacity to react in urgent circumstances, and could not capitalise on the operator's intuition and surgical experience.

It was demonstrated in [?], [?] that rapid calculation of the robot workspace to create off-line dense path-plans for intra-operative use can be achieved in minutes when using a distributed multi-node computation approach. Subsequently, it was shown that multi-threaded optimization on a single computer allows on-line inverse-kinematics for stable and collision-free paths. Nevertheless, multi-node cloud computing resources accessed through an internet connection cannot guarantee to be free of malicious software, or fully resilient to attacks (*e.g* distributed denial of service (DDoS) attack), which can compromise patient safety in a hospital environment. Therefore, local computation with dedicated computer hardware embedded in the hospital or operating theatre is more practical for the clinical translation of the developed methodologies.

Motivated by the challenges of local computation and transparent guidance this paper presents:

- A computing architecture with optimized memory management to perform pre-operative and intra-operative computations on a single device by exploiting the capabilities and power efficiency of Graphic Processing Units (GPUs)[1].
- A methodology to penalise the columns of the CTR Jacobian that lead to the robot dangerously close to the anatomy, unstable configurations, or joint limits.
- An implicit active constraint guidance scheme, which is based on the analysis of the safe and dexterous workspace of the CTR, see Fig. 1.

All contributions are evaluated via computing benchmarks and a simulated surgical intervention that compares existing guidance techniques with the presented approach.

II. Workspace Analysis — Dexterity Measure

Manipulability or dexterity analysis are standard techniques that quantify a robot's capabilities within its workspace. Operating a robot in high-dexterity regions is often critical for the successful execution of a task [**?**]. The analysis of the robot's workspace can be used to:

- Plan manoeuvres in the task-space to manipulate within regions of high dexterity.
- Optimize the joint configuration to maximize robot dexterity or steer an inverse kinematic solver towards high-dexterity configurations.

Both aspects affect autonomous robot manipulation and telemanipulation. In the latter, the dexterity analysis informs the decision-making of the robot operator.

Two common measures for dexterity are the manipulability ($\mathcal{M}$) [**?**] and the inverse condition number ($\bar{\mathcal{C}}$):

$$\mathcal{M} = \sqrt{|\mathbf{J}^e\, \mathbf{J}^{eT}|}\,, \tag{1}$$

$$\bar{\mathcal{C}} = \frac{s_{\min}}{s_{\max}}, \tag{2}$$

where $\mathbf{J}^e$ is the end-effector Jacobian, and $s_{\max}$, $s_{\min}$ are the largest, and smallest, singular values of $\mathbf{J}^e$, respectively.

[1]There are also other types of energy efficient computing accelerators *e.g.* many core processors (MIC), field-programmable gate arrays (FPGA), which my be used in lieu of a GPU.

To obtain a more accurate analysis of the manoeuvrability of a robot configuration, additional approaches have been used, *e.g.* penalization by $\mathcal{P}^q$ for joint values ($\mathbf{q} \in \mathbb{R}^{n_q}$, n_q: number of joint values) that are close to their limits ($\mathbf{q}_{\max}$, $\mathbf{q}_{\min}$) [**?**]:

$$\mathcal{P}^q = 1 - \exp\left\{-\kappa_q \prod_{i=1}^{n_q} \frac{(q_i - q_{i,\min})(q_{i,\max} - q_i)}{(q_{i,\max} - q_{i,\min})^2}\right\}, \tag{3}$$

$$\mathcal{D} = \mathcal{P}^q\, \mathcal{M}, \tag{4}$$

where κ_q is a scaling parameter, and $\mathcal{D}$ the joint-limit penalized dexterity measure.

This approach, however, does not account for redundant robots. For example, if a single joint is at its limit, then $\mathcal{P}^q$ becomes zero even if a redundant robot would continue to be manoeuvrable as the joints-space DoF (n_q) remain larger or equal to the task-space DoF (n_t). Therefore, there is a need for a measure which acts directly on the Jacobian to represent joints that are free to move. Further, the measure should be able to cope with additional programmatically imposed constraints.

This problem is addressed in [**?**] by investigating a multitude (2^{n_t}) of local Jacobians representing the possible directions of motion. The Jacobians are evaluated by computing the inverse condition number ($\bar{\mathcal{C}}$) for all $n_t \cdot 2^{n_t}$ singular values. Singular-value-decompositions (SVDs), however, are computationally expensive, and an alternative approach is required for real-time intraoperative use in redundant flexible robots.

Therefore, a penalization methodology is proposed in which the Jacobian is column-wise modified to account for joint and obstacle constraints. Instability of the CTR can also be considered based on the metrics developed in [**?**]. Viewing the Jacobian-column ($\mathbf{j}^e_i$) as a measure of the influence that a joint (q_i) has on the end-effector pose ($\mathbf{T}^e$), each column is penalized to individually account for joint-limit constraints. This leads to the formalization of the joint-limit specific penalization term ($\mathcal{P}^q_i$) as:

$$\mathcal{P}^q_i = \frac{1 - \exp\left\{\frac{4\,\kappa_q\,(q_i - q_{i,\min})(q_{i,\max} - q_i)}{(q_{i,\max} - q_{i,\min})^2}\right\}}{1 - \exp\{\kappa_q\}} \tag{5}$$

The factor «4» and the denominator «$1 - \exp\{\kappa_q\}$» in (5) are needed to normalize the penalization term such that $\mathcal{P}^q_i$ spans the interval $[0, 1]$. $\mathcal{P}^q_i$ evaluates to zero at the limits of q_i and evaluates to one in the neutral position. The scaling coefficient κ_q dictates the shape of $\mathcal{P}^q_i$ between these points.

A complementary measure penalizes joints that are responsible for collisions with the anatomy. Let $|\mathbf{d}^c|$ be the minimum distance between robot and anatomy, such that point $\mathbf{p}^c_r$ on the robot and point $\mathbf{p}^c_a$ on the anatomy define $\mathbf{d}^c$ as $\mathbf{d}^c = \mathbf{p}^c_a - \mathbf{p}^c_r$. Let the Jacobian at $\mathbf{p}^c_r$ be $\mathbf{J}^c$. These variables are used to compute the joint-specific penalization term accounting for collision constraints ($\mathcal{P}_i^c$) as:

$$\mathbf{j}^{3,c}_{\text{rel},i} := \mathbf{j}_i^{3,c} \oslash \mathbf{j}_i^{3,e}, \tag{6}$$

$$d^c_{\text{rel},i} := \begin{cases} 0 & ,\text{if} \|\mathbf{d}^c\| < d_{\min} \\ 1 & ,\text{if} \|\mathbf{d}^c\| > d_{\max} \\ \frac{\|\mathbf{d}^c\| - d^c_{\min}}{d^c_{\max} - d^c_{\min}} \Big/ \left|\left\langle \hat{\mathbf{d}}^c, \mathbf{j}^{3,c}_{\text{rel},i} \right\rangle\right|, & \text{else} \end{cases} \tag{7}$$

$$\mathcal{P}_i^c = \frac{1 - \exp\left\{\kappa_c \, d^c_{\text{rel}}\right\}}{1 - \exp\left\{\kappa_c\right\}} \tag{8}$$

In (6) $\mathbf{j}_i^{3,e}$, and $\mathbf{j}_i^{3,c}$, denote the first three elements of the i^{th} column of Jacobian $\mathbf{J}^e$, and $\mathbf{J}^c$, respectively. The Operator $\oslash$ performs component-wise division. The resulting variable $\mathbf{j}^c_{\text{rel},i}$ represents the relative motion of $\mathbf{p}_r^c$ based on the end-effector motion, which is dictated by the user[2]. For distances less than $d^c_{\min}$, and greater than $d^c_{\max}$, the relative collision distance $d^c_{\text{rel},i}$ becomes zero, and one, respectively. The scalar product in the denominator of (7) represents how the i^{th} joint is affecting the approach of the robot body to the obstacle. The nominator depends on how close the robot is to the obstacle.

The columns of the intermediate constrained Jacobian $\mathbf{J}^{q,c}$ are calculated as:

$$\mathbf{j}_i^{q,c} = \mathcal{P}_i^c \, \mathcal{P}_i^q \, \mathbf{j}_i^e \tag{9}$$

Therefore (9) imposes constraints arising both from joint-limits and the potential of causing a collision with the anatomy.

The final penalization term quantifies the possibility of the robot becoming unstable. The degree of stability d^s of the CTR, which describes the angular distance to an unstable configuration [**?**], is augmented as the non-joint-specific stability penalization term $\mathcal{P}^s$:

$$d^s_{\text{rel}} := \max\left\{\min\left\{\frac{d^s - d^s_{\min}}{d^s_{\max} - d^s_{\min}}, 1\right\}, 0\right\} \tag{10}$$

$$\mathcal{P}^s = \frac{1 - \exp\left\{\kappa_s \, d^s_{\text{rel}}\right\}}{1 - \exp\left\{\kappa_s\right\}}, \tag{11}$$

where $d^s_{\min}$ is the minimum acceptable degree of stability and $d^s > d^s_{\max}$ leads to no penalization. The final constrained Jacobian $\mathbf{J}^\star$ is calculated as:

$$\mathbf{J}^\star = \mathcal{P}^s \, \mathbf{J}^{q,c} \tag{12}$$

It is worth noting that a joint-specific stability penalization is also possible. However, due to the increased computational cost to determine the joint-specific influence on the stability, global penalization was chosen. The resulting dexterity measure is calculated in accordance to (1) as:

$$\mathcal{D} = \sqrt{\left|\mathbf{J}^\star \, \mathbf{J}^{\star T}\right|}, \tag{13}$$

and will be used to generate implicit dexterity guidance.

To compute all necessary metrics for $\mathcal{D}$, GPUs were used in the effort to create a safe inverse kinematic and guidance scheme that runs on commodity computers. The computation-architecture specifics are described next.

[2]The contribution of q_1 to the velocity of $\mathbf{p}_r^c$ *e.g.* in the x-direction is calculated according to: $\{j^e_{1,1} = \frac{\partial \dot{x}^e_1}{\partial q_1}, j^c_{1,1} = \frac{\partial \dot{x}^c_1}{\partial q_1}\} \Leftrightarrow \partial \dot{x}^c_1 = \partial \dot{x}^e_1 \frac{j^c_{1,1}}{j^e_{1,1}} \Leftrightarrow \partial \dot{x}^c_1 = \partial \dot{x}^e_1 \, j^c_{\text{rel},1,1}$

III. Concentric Tube Robot Kinematics on GPUs

The developed architecture uses the GPU to pre-operatively calculate the dexterity maps. Subsequently, intraoperatively, the multi-threaded optimisation from [**?**] is used to solve for the inverse kinematics of the robot. The optimizers are steered towards highly dexterous configurations by providing initial configurations extracted from the computed dexterity map $\mathcal{D}_{\mathcal{V},\mathcal{Q}}$, introduced in Sec. IV. This pruned dexterity map $\mathcal{D}_{\mathcal{V},\mathcal{Q}}$ is then used to visualise relevant maneuverability information.

Solving robotic kinematic and optimization problems on the GPU is non-trivial. Even though, similar to the CPU, the GPU has multiple levels of memory types, akin to *RAM*, *Cache*, *Register* with different access speeds, in GPU programming there is no automated way to optimize the use of the memory. The software architecture must provide details on the usage and layout of different types of memory. Furthermore, the GPU is not suitable for conditional statements, and random or non-coalescence memory access that reduces performance. Therefore, a dedicated memory layout for CTR kinematics computations was developed, and an heterogeneous computing approach that takes advantage of the complementary strengths of CPU and GPU.

A. GPU Memory Layout

The two main low-level programming languages for parallel computation are CUDA™, and OpenCL™. OpenCL terms are used in this paper, since OpenCL is not exclusive to GPU computing. There are equivalent terms in the CUDA framework.

The different types of memory on the GPU are: `private`, `local`, `constant`, and `global` memory. The usage of `private`, `local`, and `constant` memory has particular speed advantages. However, the memory size is limited and `constant` memory is read-only. Furthermore, even if there is no clear limit for `private` memory, it is advisable to ensure that the registers of the GPU-compute-elements can hold all `private` memory. Slow *scratch registers* are used otherwise, affecting performance.

To calculate the CTR shape, stability measure, and Jacobian matrices the memory layout was chosen as:

- `private`: helper variables, temporaries of very frequent use. Only arrays of a few elements.
- `local`: state variables of the CTR which are frequently used. Short arrays based on CTR tube and section count.
- `constant`: design parameters of the CTR that do not vary during the kinematic calculations.
- `global`: input, output variables and buffer for infrequently used CTR state variables. Large buffers storing data of each discretisation/sample point.

The memory layout in Table I details the proposed concepts. Although `local` memory can be used as a locally shared memory resource, in the proposed structure every compute element is working on independent memory areas, which are accessed coalesced. The `constant` memory stores the CTR design parameters and it allows adaptation

TABLE I

MEMORY LAYOUT ON ACCELERATOR DEVICE

Variable	Type	Size	Variable	Type	Size
`Local Memory`			`Constant Memory`		
Robot			Robot		
Base Rotation	Real	1	Max Number Sample	uint	1
Per Tube			Per Section		
Sample Curvature	Real	3	Number Tubes	uint	1
Sample Alpha	Real	1	Length	Real	1
Per Section			Tube Stiffness	Real	2
Joint Values	Real	3	Tube Curvature	Real	2
Curved Interval	Real	2	Tube Radius	Real	2
			Tube Poission Ratio	Real	2
`Global Memory` - Input[1]			`Global Memory` - Results[1]		
Joint Values	Real	$1+N_{\text{sec}}$ $+N_{\text{tube}}$	Calculation of Robot Shape Per Sample Point[2]		
Jacobian Sample Index	uint	1	Transform	Real	12
			Robot Outer Radius	Real	1
`Global Memory` - Temporary[1]			Calculation of Robot End-effector		
Per Sample Point[2]			Transform	Real	12
Alpha of Innermost Tube	Real	1	Calculation of Robot Stability		
			Degree of stability	Real	1
Curvature of Innermost Tube	Real	3	Calculation of Robot Jacobian Per Jointvalue		
			End-effector	Real	3/6
			At Sample Index	Real	3/6

Real: float (4byte) or double (8byte), uint: uint32 or uint64, to match size of Real.
N_{tube}: number of CTR tubes, N_{sec}: number of CTR sections.
[1]Memory needs to be reserved for each workitem (global_work_size).
[2]Memory needs to be reserved for the maximal number of samples.

from the host device. The memory space requirements in bytes (B), based on Table I, are:

$$\mathrm{B}_{\texttt{local}} = N_{\text{local\_size}}\,(1 + 4\,N_{\text{tube}} + 5\,N_{\text{sec}})\,\text{sizeof}(\text{Real}), \tag{14}$$

$$\mathrm{B}_{\texttt{constant}} = \text{sizeof}(\text{uint}) + N_{\text{sec}}\,(9\,\text{sizeof}(\text{Real}) + \text{sizeof}(\text{uint})), \tag{15}$$

$$\mathrm{B}_{\texttt{global\_tmp}} = 4\,N_{\text{global\_size}}\,N_{\text{max\_sample}}\,\text{sizeof}(\text{Real}), \tag{16}$$

where Real is the floating point type (float, double), $\mathrm{B}_{\texttt{local}}$ is the `local` memory size, $\mathrm{B}_{\texttt{constant}}$ the `constant` memory size and $\mathrm{B}_{\texttt{global\_tmp}}$ the `global` memory size for the temporary variables. The sizeof-function provides the byte size of the respective data type. N_{tube} is the number of tubes, and N_{sec} the number of sections the CTR comprises. $N_{\text{local\_size}}$ is the work-group size, $N_{\text{global\_size}}$ the global work size, and $N_{\text{max\_sample}}$ the maximum number of sample/discretization points along the robot centreline.

B. Heterogeneous Computing for CTR Kinematics

For the workspace analysis presented in Sec. II the heterogeneous computing approach as outlined in Fig. 2 is used. The use of the host (CPU) in step I.), and II.) is motivated by step II.) which requires the robot shape for the collision detection and proximity calculation. It is particularly efficient for proximity queries to use data structures like k-d trees. However, these data structures perform poorly on GPUs since they require fast non-coalesced random access to memory, which performs better on CPUs. Since the shape calculation in step I.) requires transfer of large amounts of memory from the device to the host, performing those calculations on the host is more efficient (as verified by experiments). Step V.) is performed on the host since it requires a singular-value-decomposition (SVD), which was not implemented on the device. Steps II.) and III.) were performed on the GPU. The memory transfer for these two steps is marginal in comparison to the computational cost. Furthermore, asynchronous *write*, *compute*, *read*, was used to fully utilize the GPU.

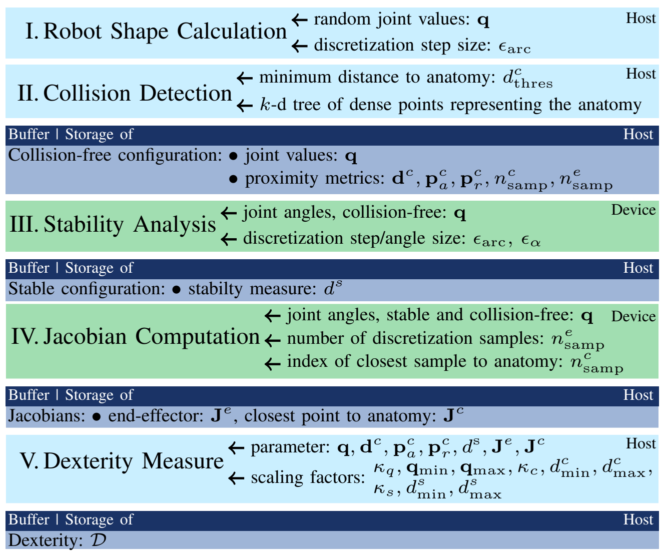
I. Robot Shape Calculation
← random joint values: $\mathbf{q}$
← discretization step size: ϵ_{arc}
Host
II. Collision Detection
← minimum distance to anatomy: d^c_{thres}
← k-d tree of dense points representing the anatomy
Host
Buffer | Storage of
Host
Collision-free configuration: ● joint values: $\mathbf{q}$
● proximity metrics: $\mathbf{d}^c, \mathbf{p}^c_a, \mathbf{p}^c_r, n^c_{\text{samp}}, n^e_{\text{samp}}$
III. Stability Analysis
← joint angles, collision-free: $\mathbf{q}$
← discretization step/angle size: $\epsilon_{\text{arc}}, \epsilon_\alpha$
Device
Buffer | Storage of
Host
Stable configuration: ● stabilty measure: d^s
IV. Jacobian Computation
← joint angles, stable and collision-free: $\mathbf{q}$
← number of discretization samples: n^e_{samp}
← index of closest sample to anatomy: n^c_{samp}
Device
Buffer | Storage of
Host
Jacobians: ● end-effector: $\mathbf{J}^e$, closest point to anatomy: $\mathbf{J}^c$
V. Dexterity Measure
← parameter: $\mathbf{q}, \mathbf{d}^c, \mathbf{p}^c_a, \mathbf{p}^c_r, d^s, \mathbf{J}^e, \mathbf{J}^c$
← scaling factors: $\kappa_q, \mathbf{q}_{\text{min}}, \mathbf{q}_{\text{max}}, \kappa_c, d^c_{\text{min}}, d^c_{\text{max}}, \kappa_s, d^s_{\text{min}}, d^s_{\text{max}}$
Host
Buffer | Storage of
Host
Dexterity: $\mathcal{D}$

Fig. 2. Heterogeneous computing approach running different computation tasks on the host (CPP) and accelerator devices (OCL). Inputs for each task/algorithm are indicated with: $\leftarrow$. Outputs are stored in buffers, which are listed below the respective algorithm.

All dexterity metrics for a given configuration are calculated as described in Fig. 2 and stored to be used as implicit constraints.

IV. IMPLICIT CONSTRAINTS

To use the calculated Dexterity ($\mathcal{D}$) in task-space, a discretized workspace bounding box $\mathcal{V}$, comprised of voxels ($v \in \mathbb{R}^3$), is generated. $\mathcal{D}$ depends on the robot configuration, which is defined by the joint values ($\mathbf{q}$). Different sets of joint values can map to the same end-effector position in task-space ($\mathbf{x}$). Therefore, considering a discretized/voxelized workspace, each configuration ($\mathbf{q}$) with dexterity ($\mathcal{D}$) and position ($\mathbf{x}$) maps to a voxel (v), with i, j, k denoting the indices of the voxel, which also includes position $\mathbf{x}$:

$$\mathbf{q} \rightarrow \{\mathcal{D}(\mathbf{q}), \mathbf{x}(\mathbf{q})\} \rightarrow v(i, j, k) \tag{17}$$

To provide visual cues to the operator on how to plan the robot manipulation to a certain target point, the maximum dexterity value in a voxel ($\mathcal{D}_\mathcal{V}$) is calculated as:

$$\mathcal{D}_\mathcal{V}(i, j, k) = \max_{\mathbf{q}_m} \{\mathcal{D}(\mathbf{q}_m) \mid \mathbf{x}(\mathbf{q}_m) \in v(i, j, k)\} \tag{18}$$

This voxel dexterity ($\mathcal{D}_\mathcal{V}$) is provided to a volume rendering engine that visualizes the regions of the workspace with highest manoeuvrability. These should be considered in the user's path planning to safely reach the target, see Fig. 1. Hence, the user is provided with an implicit description of the robot's capabilities, which can be combined with other factors such as experience or task-specific considerations that

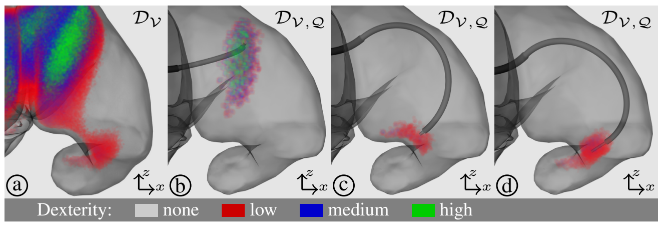
$\mathcal{D}_{\mathcal{V}}$
$\mathcal{D}_{\mathcal{V},\mathcal{Q}}$
$\mathcal{D}_{\mathcal{V},\mathcal{Q}}$
$\mathcal{D}_{\mathcal{V},\mathcal{Q}}$
a
b
c
d
z
x
Dexterity: none low medium high

Fig. 3. Visualization of CTR dexterity, a: maximal dexterity based on tip position, b, c, d: maximal dexterity considers also current joint values.

are difficult to be formalized and might be operator- and patient-specific.

Furthermore, during task execution, a current-configuration-specific dexterity map $\mathcal{D}_{\mathcal{V},\mathcal{Q}}$ based on current joint values ($\mathbf{q}$) and voxel (v) is generated and updated:

$$\mathcal{D}_{\mathcal{V},\mathcal{Q}}(\mathbf{q},v) = \max_{\mathbf{q}_m}\{\mathcal{D}(\mathbf{q}_m)\,|\,\mathbf{x}(\mathbf{q}_m) \in v \wedge \tau_q(\mathbf{q},\mathbf{q}_m)\} \quad (19)$$

where τ_q is a threshold function:

$$\tau_q(\mathbf{q}',\mathbf{q}'') := \begin{cases} 1, \text{if } |q_i' - q_i''| < q_i^{\text{thres}} \; \forall i \in \{1 \dots n_q\} \\ 0, \text{else} \end{cases}, \quad (20)$$

where $\mathbf{q}', \mathbf{q}''$ are two sets of joint values, and $\mathbf{q}^{\text{thres}}$ is the upper threshold of acceptable joint-value differences. Therefore, dexterity values stemming from configurations highly different to the current configuration are filtered out. A visualisation of $\mathcal{D}_{\mathcal{V}}$ is depicted in Fig. 3 (pre-operative view), and Fig. 3b, c, d shows the configuration specific voxel dexterity $\mathcal{D}_{\mathcal{V},\mathcal{Q}}$ updated during robot manipulation (intra-operative view).

To support the user to manipulate the CTR with high maneuverability, haptic guidance forces attract the operator to regions of high dexterity. The guidance forces are calculated as:

$$F(\mathbf{q},v) = \kappa_f \sum_i \frac{\mathbf{x}(v_i) - \mathbf{x}(v)}{\|\mathbf{x}(v_i) - \mathbf{x}(v)\|^2} \big(\mathcal{D}_{\mathcal{V},\mathcal{Q}}(\mathbf{q},v_i) - \mathcal{D}_{\mathcal{V},\mathcal{Q}}(\mathbf{q},v)\big) \Big| \|\mathbf{x}(v_i) - \mathbf{x}(v)\| < x_F^{\text{thres}}, \quad (21)$$

where $\mathbf{x}(v)$ provides the position of a workspace vertex v, x_F^{thres} is distance threshold for considering voxel dexterity, and κ_f is a force scaling factor.

V. Experiments

To verify the efficiency of the presented heterogeneous computing approach and the effectiveness of the dexterity-based guidance, two experiments were conducted. The first assesses the computational speed improvements using GPUs, and the second assesses the visual and haptic guidance scheme proposed.

A. Benchmarking Experiment

The implementation was tested using a CTR with 3 sections and 4 tubes. The robot parameters are listed in Table II. An AMD FirePro™ W9100 GPU was used as OpenCL device. It features 32 kB `local` memory and

TABLE II
Robot Parameters (3 Sections, 4 Tubes)

Robot Section Type	**Curvature [mm^{-1}]**	**Straight/Curved Length [mm]**	**Stiffness Ratio**	**Tube Index**
variable curvature	0.0522	0.00/66.47	5:1	1, 2
fixed curvature	0.0526	66.47/66.70	1:1	3
fixed curvature	0.0	113.17/12.65	0.05:1	4

64 kB `constant` memory. The preferred work-group size ($N_{\text{local\_size}}$) is a multiple of 64. Using double-precision with a size of 8 B allows to have a work-group size of 128 consuming $\text{B}_{\texttt{local}} = 32$ kB of `local` memory and $\text{B}_{\texttt{constant}} = 248$ B of `constant` memory. Furthermore, the maximal number of sample ($N_{\text{max\_sample}}$) was set to 256, which provides a sufficiently large number of samples for the targeted discretization step of $\epsilon_{\text{arc}} = 1$ mm. The number of work-groups ($N_{\text{local\_size}}$) was set to 88, which corresponds to twice the number of compute units of the GPU. Therefore, temporary `global` memory of $\text{B}_{\texttt{global\_tmp}} = 88$ MB had to be reserved. With single-precision a work-group size of 256 was used, which also results in 32 kB of `local` memory. The `constant` memory reduced to 124 B and maintaining the number of work-groups the requirements, for the temporary `global` memory of 88 MB was also maintained. Code optimization with regards to `private` memory and temporary variables resulted in the avoidance of *scratch registered*, which was verified through runtime profiling.

Benchmarking experiments where conducted using:

- CPU: 2x Intel® Xeon® E5-2637v3 with a combined thermal design power (TDP) of 270 W, and
- GPU: AMD FirePro™ W9100 GPU with a similar TDP of 275 W.

Table III shows the benchmarking results using the heterogeneous computing (HetC) with CPU and GPU (TDP=545 W) and the homogeneous computing approach (HomC) using only the CPU (TDP=270 W). To fully utilize the GPU the random numbers for the translational joint-values were chosen in intervals (*i.e.* generate random configurations which result in $[a,b]$% extended CTRs). This translation interval approach (TIA) ensures that one work-group on the GPU processes CTR of similar lengths and with similar numbers of discretisation points, which results in similar instruction paths for all work-items in a work-group. The intervals are shifted during the Monte Carlo sampling such that the CTR lengths of all joint-configurations are uniformly distributed.

It was found that the bottleneck in the conducted experiment was the CPU. Even with a single GPU, the CPU was not able to generate collision free-configurations fast enough to be then tested by the GPU for stability and Jacobian calculation. In order to show the advantage the GPU has over the CPU regarding stability and Jacobian computation, the workload was shifted to step III.) and IV.) (stability, Jacobian) by choosing only short robots (TIA $\in [10,20]$%, likely collision-free) and reducing the discretization step size ϵ_{arc} by a factor of 10. With this parameterization, a dual GPU configuration was tested (TDP=820 W).

1) Results: The results of the benchmark experiment are listed in Table III. Using only the CPU the duration of generating a dense dexterity map is 116-1070 % longer. With the TIA approach the heterogeneous approach shows an increased performance of 54 %. The two top benchmarks depicted in Table III report results of a non-fully utilized GPU. The reduction of the discretization step size increased the workload of steps III.) and IV.) and led to the doubling of the CPU computation time. In contrast, the GPU's computation time increased only by 40%. In a dual GPU setup the overall computation time using the heterogeneous approach is reduced by a factor of approximately 11 in comparison to the CPU. Furthermore, the overall energy consumption decreased in all benchmarks, when switching from CPU only to heterogeneous computation. Utilizing a single GPU fully leads to energy consumption improvements of up to 268 %.

2) Discussion: The experimental results show that using a GPU for the kinematics of CTRs within the pre-operative workspace analysis can result in significant computational speed up on a single computer. For random sampling an interval based scaling of the CTR length, which ensures similar execution paths on the accelerator proved to increase efficiency. Furthermore, it has potential to reduce the overall energy consumption since accelerators are often more power efficient. A good thermal energy efficiency is important since it affects how dense the computing performance can be packaged based on cooling constraints. The results lead to the conclusion that the computational performance of the CPU and GPU have to be tailored with regards to the workload of the individual computation steps. For this experiment, for example, a less powerful GPU would be sufficient.

B. Dexterity Guidance

The clinically relevant scenario is simulated based on a challenging procedure that involves cauterisation of the choroid plexus in hydrocephalic ventricles. During the intervention an elongated CTR needs to access the horns of the ventricles and cauterise them to reduce the production of cerebrospinal fluid and to consequently decrease ventricle pressure. The procedure is anticipated to be safer than cerebral shunts and an alternative to endoscopic third ventriculostomy [?]. It requires a curved CTR that tends to become unstable and is likely to collide with the anatomy.

TABLE III
BENCHMARK HETEROGENEOUS/ HOMOGENEOUS DEXTERITY COMPUTATION

Real Type	Duration HetC [s]	Duration HomC [s]	Energy HetC [kWs]	Energy HomC [kWS]	CPU Energy Surplus
$4.19 \cdot 10^6$ Random Samples, 1×GPU, without TIA, ϵ_{arc}= 1 mm					
float	261.4	566.9	142.5	153.1	7.4%
double	262.0	599.8	142.8	162.0	13.4%
$4.19 \cdot 10^6$ Random Samples, 1×GPU, with TIA, ϵ_{arc}= 1 mm					
float	169.7	565.6	92.5	152.7	65.1%
double	170.9	624.1	93.1	168.5	80.9%
$4.19 \cdot 10^6$ Random Samples, 1×GPU, TIA $\in [10, 20]$%, ϵ_{arc}= 0.1 mm					
float	242.2	1798.9	132.0	485.7	267.9%
double	546.6	2098.0	297.9	566.5	90.1%
$4.19 \cdot 10^6$ Random Samples, 2×GPU, TIA $\in [10, 20]$%, ϵ_{arc}= 0.1 mm					
float	154.2	1804.2	126.4	487.1	285.2%
double	294.7	2095.5	241.7	565.8	134.2%

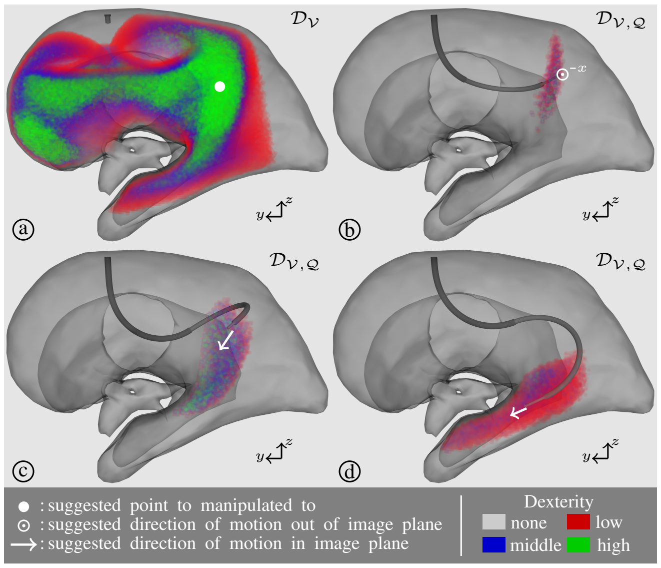
$\mathcal{D}_{\mathcal{V}}$
$\mathcal{D}_{\mathcal{V},\mathcal{Q}}$
$-x$
y z
a
b
c
d
● : suggested point to manipulated to
⊙ : suggested direction of motion out of image plane
⟶ : suggested direction of motion in image plane
Dexterity
none
low
middle
high

Fig. 4. Visual guidance based on dexterity analysis. Information of the robot's dexterity is provided to the user to enable him/her in combination with experience to plan a stable path way. a.) pre-operative view, b, c, d.) intra-operative view.

Eleven user experiments were conducted in which the participants were asked to manoeuvre the CTR to 13 cauterization locations within the ventricle anatomy. In each experiment the user was asked to complete the task in three different modes:

- «no planning»: solely using on-line inverse kinematics, no provision of haptic or visual cues, described in [?].
- «path planning»: provision of assists to the user with a guidance path ensuring stable configurations when followed, described in [?].
- «dexterity planning»: guiding the user through visual and haptic feedback to manipulate within regions of high dexterity, presented in Sec. IV.

An example view of the visual guidance provided by the «dexterity planning» mode is depicted in Fig. 4. The visual guidance scheme provided the users with dexterity information, which helped to plan a path based on anatomical constraints and robot capabilities. This approach does not explicitly generate a guidance path. The order of execution of the three modes are alternated between users to eliminate learning effects.

1) Results: The quantitative evaluation criteria is the task duration, depicted in Fig. 5. It shows that the modes using path-, and dexterity planning, enable the user to finish the task faster. The results show no statistically significant difference between the task duration when using «path planning» or «dexterity planning». However, both planning methods show a statistically significant difference in the duration to the «no planning» mode (based on a significance-level of 5%,

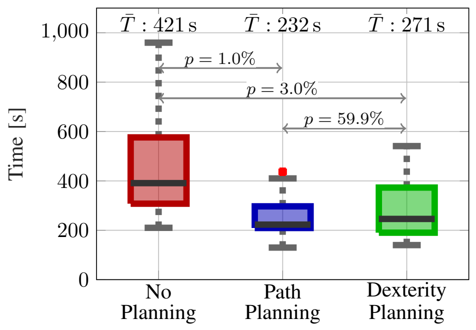

Fig. 5. Statistical analysis of experiment task durations. $\bar{T}$: interquartile mean duration [s], p: p-value of two-sided Wilcoxon rank sum test [%].

using the two-sided Wilcoxon rank sum test). Calculating the interquartile mean using the «no planning» mode the users were 81% and 55% slower compared to «path planning» and «dexterity planning», respectively.

The qualitative assessment was performed by users rating their perception of the three modes on a seven-level Likert scale regarding: i.) overall performance, ii.) frustration, iii.) haptic guidance, and iv.) visual guidance. A negative value encodes a undesired perception, and a positive rating a favoured characteristic, see Fig. 6. The qualitative assessment shows that the «no planning» mode was perceived worse in all regards. The visual guidance was perceived best in the «dexterity planning» mode and in the remaining categories the «path planning» mode was favoured.

2) Discussion: The user experiment showed that guidance approaches are effective for CTR manipulation and help with unintuitive manipulation of CTRs. The «path planning» implementation from [**?**] performs a little better than the presented «dexterity planning» approach. In particular the haptic feedback was criticized as only partially useful, and increased user frustration. The haptic feedback in the implementation of this paper always pulls towards high dexterity regions, although it might be necessary to enter lower dexterity regions to reach the target. An additional processing step based on $F(\mathbf{q}, v)$ and motion redirection as presented in [**?**], might overcome this static force problem. Nevertheless, the presented dexterity based approach has multiple advantages. The user can better account for:

- non-modelled factors (*e.g.* additional safety margins within the anatomy),

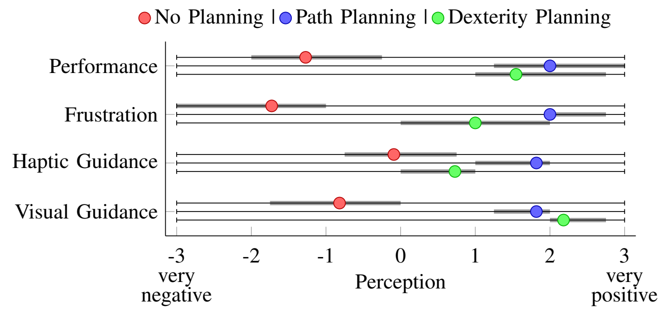

Fig. 6. User perceptional assessment of the different modes of operation, marker represent mean value. Positive values represent favored perception.

- calibration and registration errors,
- surgical experience,
- unexpected situations.

Furthermore, it could be used in addition to the «path planning» approach or as preceding step to provide the operator with an initial understanding of the robot capabilities within the respective intervention.

VI. Conclusion

This paper presented the use of implicit active constraints for concentric tube robots based on the analysis of the safe and dexterous workspace to rapidly inform the operator with visual and haptic cues about the global and configuration-specific manoeuvrability of the robot. A heterogeneous computing architecture was presented, which reduced computation time and increased energy efficiency. This computing approach was further used to generate a dexterity guidance map, which proved to perform equally well to path guidance approaches, and thereby provided the user with more freedom on their robot manipulation. Further, improvements in the visual display and haptic rendering will provide more transparent guidance and are anticipated to result in a more intuitive user experience.

VII. Acknowledgement

The authors gratefully acknowledge Prof. Pierre Dupont from the Paediatric Cardiac Bioengineering Lab of Boston Children's Hospital, Harvard Medical School for providing the anatomy used in this paper.