Inverse Kinematics Control Methods for Redundant Snakelike Robot Teleoperation During Minimally Invasive Surgery

Pierre Berthet-Rayne, Konrad Leibrandt, Gauthier Gras, Philippe Fraisse, André Crosnier, Guang-Zhong Yang

Inverse Kinematics Control Methods for Redundant Snake-Like Robot Teleoperation during Minimally Invasive Surgery

Pierre Berthet-Rayne[1], Konrad Leibrandt[1], Gauthier Gras[1],
Philippe Fraisse[2], André Crosnier[2], and Guang-Zhong Yang[1] *fellow IEEE*

***Abstract*—The real-time teleoperation or telemanipulation of redundant snake-like robots for minimally invasive surgery in a master-slave configuration is a complex problem. There are many possible mappings between a master's standard 6 degrees of freedom and a redundant slave robot, typically with $n >> 6$ degrees of freedom. This paper introduces a snake-like robot for ear, nose and throat surgery. The robot's architecture is comprised of $n = 26$ joint variables. Six different control methods were investigated. The methods are compared through simulation with a user study. Each participant performed the same task using each of the six different control methods. Based on the metrics selected, the sparse pseudo-L_0 and our proposed approach performed better in terms of intuitiveness, real-time capabilities and overall occupied volume.**

***Index Terms*—Medical Robots and Systems, Surgical Robotics: Laparoscopy, Telerobotics and Teleoperation, Redundant Robots, Tendon/Wire Mechanism**

I. INTRODUCTION

ADVANCES in medical robots are heading toward flexible robotic devices that can follow anatomical pathways to perform surgery endoluminally or without the need for incisions [1], [2]. Such medical robots are often composed of multiple sections that can bend in various shapes like a snake. In existing surgical workflows, such robots would be teleoperated by the surgeon to navigate through the anatomy and reach surgical sites of interest. However, the teleoperation mapping between the surgeon's hand motion and the robot's multiple articulations in a Master-Slave configuration is a complex task, especially in the case of redundant robots.

Multiple approaches exist to address the problem of navigation and control of snake-like robots. Approaches may include either hardware or software or both. Hardware approaches consist of designing a device with the intrinsic hardware capabilities to follow the anatomy; this includes passive-active devices combining standard flexible endoscopes with robotic actuation [3] or hardware-based follow the leader navigation combined with shape locking [4]. Software-based approaches aim at providing an intuitive method to control a redundant robot with complex kinematics with a standard 6 to 7 degrees of freedom (DOF) master interface. Software approaches include inverse kinematics, motion planning and machine learning. Motion planning approaches have the main advantage that they provide obstacle avoidance together with navigation [5]. However, such methods are often computationally expensive, especially if a $n \geq 6$ DOF workspace is considered, and may require pre-computation in order to be used in real-time [6]. Inverse kinematics based methods usually address the problem of control for known pre-set trajectories where real-time capabilities are optional. Teleoperation, however, requires additional constraints for successful control. In the field of surgical robotics, teleoperation is a common mode of operation as it allows the surgeon to be part of the control loop. During the teleoperation of snake-like robots, the trajectory is unknown and is determined by the surgeon. In this context, some specific additional features need to be considered for successful control. First of all, the motion of the slave robot must match the motion of the master interface; this is necessary for the control to be intuitive. A second parameter to consider is the speed to find a solution to match the master interface. The solver must be fast enough that there is no lag in the control loop. Finally, in the specific case of redundant snake-like robots and minimally invasive surgery, the robot motion must be minimal or sparse so as to

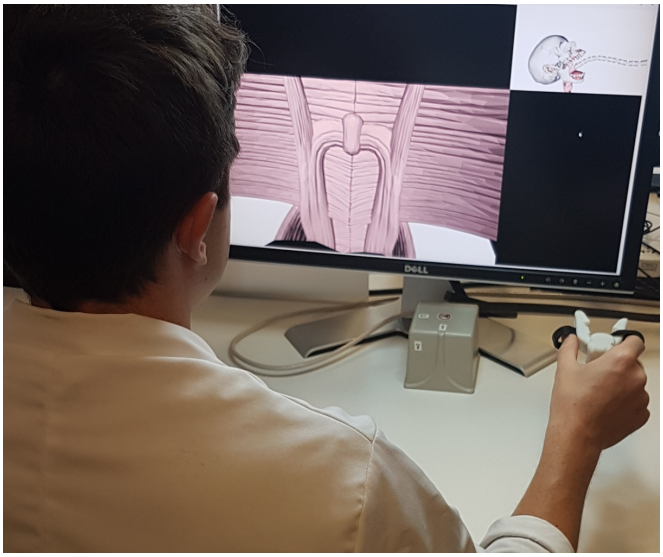

Fig. 1: Simulation of an ENT procedure. The user is sitting in front of the simulator and uses the hand-held master interface to perform the simulated navigation tasks.

[1]P. Berthet-Rayne, K. Leibrandt, G. Gras and G.-Z. Yang are with the Hamlyn Centre for Robotic Surgery, Imperial College London, London, United Kingdom ptb14 at imperial.ac.uk

[2] A. Crosnier and P. Fraisse are with the Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM), CNRS-University of Montpellier, Montpellier, France

minimize the impact on the patient's anatomy.

The aim of this paper is to evaluate existing inverse kinematic methods based on the three key features presented above: being intuitive, fast and without colliding with the anatomical boundary. The robot and each method are applied to a specific clinical specialty: Ear Nose Throat surgery (ENT). The proposed robot is named as follow: the Intuitive Imaging Sensing Navigated and Kinematically Enhanced robot: the *i²Snake*. The *i²Snake* was designed to perform ENT surgery through the mouth cavity, see Fig. 1. This paper investigates the feasibility of using existing and modified inverse kinematics methods [7], [8], [9] to teleoperate snake-like robots during ENT procedures such as tonsil removal, general endoscopic inspection, and tumor resection.

The paper is organized as follow: section II introduces a snake-like robot architecture. The robot's joint model and DH parameters are analyzed and summarized in the Denavit-Hartenberg (DH) table II. Section III introduces a set of six inverse kinematics control methods that were selected to be implemented on the robot. The section IV presents the results of a teleoperation user experiment that was performed on a virtual simulator. Finally section V concludes on the results and the key features required for successful control of snake-like robots in surgical environments.

II. ROBOT MODELING

A. Rolling-Joint Modeling

The proposed robot is composed of multiple serial rolling-joints. The rolling-joint is a bio-inspired type of articulation where two surfaces roll against each other. This type of joint has many advantages over conventional revolute joints such as low friction, symmetric actuation, large force transmission, but one of the drawback is its modeling complexity [10]. As the two surfaces roll against each other, the contact point moves, resulting in a combined rotation and translation motion. To model this behavior in terms of DH parameters, a single rolling-joint can be subdivided into two revolute joints that will move with the same angle $\theta/2$ as presented in [11]. As such, each rolling-joint will result in two revolute joints as shown in the DH Table I and Fig. 2. Each rolling-joint provides 1 DOF, so to obtain a 2 DOF robot, the DH table will have 4 joints, hence doubling the model complexity. By arranging multiple rolling-joints orthogonally, it is possible to create a redundant robot that can move in any directions.

B. The i²Snake Virtual Robot

1) Robot Model: The i²Snake Robot is a snake-like robot designed for ENT surgery. It is composed of a serial chain of rolling-joints. It is a redundant robot with 8 independent DOF.

TABLE I: DH table of a single rolling-joint.

i	$\mathbf{a}$	α	d	θ
1	a_1	0	0	$\theta_1/2$
2	a_1	0	0	$\theta_1/2$

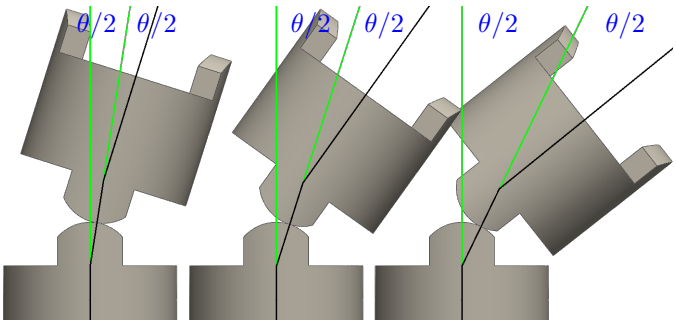

Fig. 2: Rolling-joint model. As the joints roll against each other, the contact point is moving along the curved surface resulting in a combined rotation and translation motion. This behavior can be modeled using two standard revolute joints.

The first joint is prismatic and allows for the translation motion of the whole robot. The second joint is revolute and rotates the entire snake robot along its longitudinal axis. The following 12 joints are rolling-joints arranged orthogonally, allowing to bend in any direction. Let q be the vector composed of the joint variables of the mechanical structure:

$$q = (q_1, q_2, \ldots, q_{26})^T \tag{1}$$

To reduce the amount of motors required to actuate the robot, the rolling-joint are coupled mechanically by pairs rotating in the same direction, starting from the base. Hence, the movement of the robot is controlled through a set of 8 independent control variables ξ:

$$\xi = (\xi_1, \xi_2, \ldots, \xi_8)^T \tag{2}$$

The correspondences between q_i and ξ_i are specified in the DH parameters table II, where a_1 and a_2 are constant parameters: $a_1 = 0.00618$ m and $a_2 = 0.01182$ m. The relations between q_i and ξ_i are the following:

$$q_1 = \xi_1 \tag{3}$$
$$q_2 = \xi_2 \tag{4}$$
$$(q_3, \ldots, q_{10})^T = G_1\xi_3 + G_2\xi_4 \tag{5}$$
$$(q_{11}, \ldots, q_{18})^T = G_1\xi_5 + G_2\xi_6 \tag{6}$$
$$(q_{19}, \ldots, q_{26})^T = G_1\xi_7 + G_2\xi_8 \tag{7}$$

where

$$G_1 = \begin{pmatrix}0.5 & 0.5 & 0 & 0 & 0.5 & 0.5 & 0 & 0\end{pmatrix}^T \tag{8}$$
$$\text{and } G_2 = \begin{pmatrix}0 & 0 & 0.5 & 0.5 & 0 & 0 & 0.5 & 0.5\end{pmatrix}^T$$

The first two joints are directly controlled. The other joints are grouped into three sections: proximal, middle and distal. Each section being associated with two control variables. Finally, it yields

$$q = G \cdot \xi \tag{9}$$

where G is a 26 x 8 matrix expressed as

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & G_1 & G_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & G_1 & G_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & G_1 & G_2 \end{pmatrix} \tag{10}$$

The resulting architecture of the i^{2}Snake is represented in the DH Table II.

2) Tool Frame: The pose of the tool in the local frame 26 is defined by the following homogeneous matrix:

$$
\begin{aligned}
{}^{26}T_{tool} &= \\
&= \begin{pmatrix} 0 & 0 & 1 & a_3 + a_2 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0.043 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}
\end{aligned} \tag{11}
$$

where $a_3 = 0.0312\ m$

3) Forward Kinematics: The computation of the forward kinematics is straightforward. The pose of the tool in the base frame is obtained from the product of the homogeneous matrices computed from the DH parameters.

$$
\begin{aligned}
{}^{0}T_{tool} &= \begin{pmatrix} A_{tool} & p_{tool} \\ 0 & 1 \end{pmatrix} = \\
&= {}^{0}T_1(q_1) \cdots {}^{25}T_{26}(q_{26}) {}^{26}T_{tool}
\end{aligned} \tag{12}
$$

Given the equations (3) to (7) , q_1 to q_{26} are linear functions of the control vector ξ, and consequently it is easy to express ${}^{0}T_{tool}$ as a function of ξ.

4) Jacobian matrix: The velocity in the operational space is denoted $(v_{tool}, \omega)^T$, where $v_{tool} = \frac{dp_{tool}}{dt}$ is the linear velocity associated with the tool center point, and ω the angular velocity resulting from the relation $\dot{A}_{tool} = S(\omega)A_{tool}$. The kinematic model is obtained from the relation

$$
\begin{pmatrix} v_{tool} \\ \omega \end{pmatrix} = J_q(q)\ \dot{q} \tag{13}
$$

where J_q is the 6 x 26 Jacobian matrix of the mechanical structure defined in the base frame. The vector $(V_{tool}, \omega)^T$ can be computed by superposing the contributions of each joint in the mechanical structure. From equation (9), we have $\dot{q} = G\ \dot{\xi}$, and then the Jacobian matrix J with respect to the control vector ξ is equal to:

$$
J(\xi) = J_q(G\ \xi)\ G \tag{14}
$$

TABLE II: Extended DH table of the snake-like robot.

i	a	α	d	θ	Type	q	ξ
1	$\mathbf{0}$	0	$\boldsymbol{d_1}$	$\mathbf{0}$	P	q_1	ξ_1
2	$\mathbf{0}$	0	0	$\boldsymbol{\theta_1}$	R	q_2	ξ_2
3	$\mathbf{0}$	$\boldsymbol{\pi/2}$	0	$\boldsymbol{\theta_2}$	R	q_3	ξ_3
4	$\boldsymbol{a_1}$	0	0	$\boldsymbol{\theta_2}$	R	q_4	ξ_3
5	$\boldsymbol{a_2}$	$\boldsymbol{\pi/2}$	0	$\boldsymbol{\theta_3}$	R	q_5	ξ_4
6	$\boldsymbol{a_1}$	0	0	$\boldsymbol{\theta_3}$	R	q_6	ξ_4
7	$\boldsymbol{a_2}$	$\boldsymbol{-\pi/2}$	0	$\boldsymbol{\theta_2}$	R	q_7	ξ_3
8	$\boldsymbol{a_1}$	0	0	$\boldsymbol{\theta_2}$	R	q_8	ξ_3
9	$\boldsymbol{a_2}$	$\boldsymbol{\pi/2}$	0	$\boldsymbol{\theta_3}$	R	q_9	ξ_4
10	$\boldsymbol{a_1}$	0	0	$\boldsymbol{\theta_3}$	R	q_{10}	ξ_4
11	$\boldsymbol{a_2}$	$\boldsymbol{-\pi/2}$	0	$\boldsymbol{\theta_4}$	R	q_{11}	ξ_5
12	$\boldsymbol{a_1}$	0	0	$\boldsymbol{\theta_4}$	R	q_{12}	ξ_5
13	$\boldsymbol{a_2}$	$\boldsymbol{\pi/2}$	0	$\boldsymbol{\theta_5}$	R	q_{13}	ξ_6
14	$\boldsymbol{a_1}$	0	0	$\boldsymbol{\theta_5}$	R	q_{14}	ξ_6
15	$\boldsymbol{a_2}$	$\boldsymbol{-\pi/2}$	0	$\boldsymbol{\theta_4}$	R	q_{15}	ξ_5
16	$\boldsymbol{a_1}$	0	0	$\boldsymbol{\theta_4}$	R	q_{16}	ξ_5
17	$\boldsymbol{a_2}$	$\boldsymbol{\pi/2}$	0	$\boldsymbol{\theta_5}$	R	q_{17}	ξ_6
18	$\boldsymbol{a_1}$	0	0	$\boldsymbol{\theta_5}$	R	q_{18}	ξ_6
19	$\boldsymbol{a_2}$	$\boldsymbol{-\pi/2}$	0	$\boldsymbol{\theta_6}$	R	q_{19}	ξ_7
20	$\boldsymbol{a_1}$	0	0	$\boldsymbol{\theta_6}$	R	q_{20}	ξ_7
21	$\boldsymbol{a_2}$	$\boldsymbol{\pi/2}$	0	$\boldsymbol{\theta_7}$	R	q_{21}	ξ_8
22	$\boldsymbol{a_1}$	0	0	$\boldsymbol{\theta_7}$	R	q_{22}	ξ_8
23	$\boldsymbol{a_2}$	$\boldsymbol{-\pi/2}$	0	$\boldsymbol{\theta_6}$	R	q_{23}	ξ_7
24	$\boldsymbol{a_1}$	0	0	$\boldsymbol{\theta_6}$	R	q_{24}	ξ_7
25	$\boldsymbol{a_2}$	$\boldsymbol{\pi/2}$	0	$\boldsymbol{\theta_7}$	R	q_{25}	ξ_8
26	$\boldsymbol{a_1}$	0	0	$\boldsymbol{\theta_7}$	R	q_{26}	ξ_8

III. Teleoperation Control Methods

A. Pseudo-inverse Jacobian

The task is described by a target vector $x_d \in \mathbb{R}^m$, defined as a constant. The tracking error is estimated by the vector $e(\xi) = x_d - x(\xi)$. The kinematic control aims at finding a solution $\dot{\xi}$ of the equation $\dot{e} = \dot{x}_d - J\dot{\xi} = -J\dot{\xi}$, where e is assumed to be a solution of the (first-order) differential equation $\dot{e} + \eta e = 0$, with $\eta > 0$. The local search is expressed as the following optimization problem:

$$
\text{Find } \dot{\xi}^* \in \min_{\dot{\xi}} \quad \|J\dot{\xi} - \eta e\|_2^2 \tag{15}
$$

(15) has a solution $\dot{\xi} = J^+ \eta e$ where J^+ is the pseudo-inverse of J [12]. An alternative to this approach is the damped least squares (DLS) method that allows to avoid problems caused by singularities using the pseudo-inverse and was therefore used in the user study. DLS consists of changing the cost function defined in (15) by adding a second order regularization term (Tikhonov regularization) and in finding the value of $\dot{\xi}$ that minimizes the quantity:

$$
\text{Find } \dot{\xi}^* \in \min_{\dot{\xi}} \quad \|J\dot{\xi} - \eta e\|_2^2 + \lambda \|\dot{\xi}\|_2^2 \tag{16}
$$

where $\lambda > 0$ is a non-zero damping constant. The analytic solution of (16) is given by

$$
\dot{\xi} = J^T (JJ^T + \lambda^2 \mathbb{I})^{-1} \eta e \tag{17}
$$

B. Joint limit based Jacobian modification

The Jacobian pseudo-inverse optimization according to (15) or (16) represents an unconstrained optimization approach, which does not take constraints such as joint limits into account. For redundant robots a common approach to avoid configurations at the joint limits is to use *null*-space mapping and to use secondary goal which optimizes for joint values close to the neutral position. However, commonly that means that a configuration close to the neutral position is preferred, which might not be the case and which can result in unnecessary motion.

Considering hard joint limits in a Jacobian based approach has been proposed in [8]. The following approach addresses joint limits only when they occur by modifying J to the modified Jacobian (J_{jl}) which takes joint limits into account and is an evolution of the approach presented in [13]. The i^{th} modified Jacobian column $j_{jl,i}$ is computed based on the i^{th} column of the end-effector Jacobian j_i as:

$$
j_{jl,i} = \nu_i(\xi, \dot{\xi}, \xi_{min}, \xi_{max})\, j_i \tag{18}
$$

where the column scaling function ν_i is defined as:

$$\nu_i(\xi, \dot{\xi}, \xi_{min}, \xi_{max}) = \begin{cases} 0, \text{ if } \xi_i \leq (\xi_{min,i} - \dot{\xi}_i \delta t) \wedge \dot{\xi}_i < 0 \\ 0, \text{ if } \xi_i \geq (\xi_{max,i} - \dot{\xi}_i \delta t) \wedge \dot{\xi}_i > 0 \\ 1, \text{ else} \end{cases} \tag{19}$$

The resulting Jacobian reflects that a joint at the limit can not be actuated pass that limit, since columns at the limits are damped to zero. The column damping in (19) can alternatively also be performed smooth over the motion range. The joint velocity is then computed according to (17) with J_{jl} instead of J. In case of a redundant robot the remaining joints can compensate for the loss of DOF. In case the robot is not redundant the Jacobian will loose rank and the pseudo-inverse will provide the best compromise.

C. Sparse Pseudo L_0 Norm

This approach provides a solution $\dot{\xi}$ that is sparse, it means to find the smallest non-zero coefficients of a linear system expressed as:

$$\begin{aligned} \text{Find } \dot{\xi}^* \in \min_{\dot{\xi}} \quad & \|\dot{\xi}\|_0 \\ \text{subject to} \quad & J\dot{\xi} = \eta e \end{aligned} \tag{20}$$

with l_0-norm is a non-zero counting norm that is defined and used for compressed sensing applications in signal processing [14], [15]. This problem is a NP-hard problem [16]. Despite this, all combinations of non-zero entries can be computed in a reasonable time for a low dimensional system such as a robotic system. In addition, the research field can be reduced by assuming that the Jacobian matrix J is full row rank matrix equal to m. Thus, the number of non-zero entries $N_0 \geq m$. Consequently, the number of combinations N_c to find the optimal solution is given by:

$$\begin{aligned} \frac{n!}{m!(n-m)!} \leq N_c \leq \frac{n!}{m!(n-m)!} + \\ + \frac{n!}{(m+1)!(n-(m+1))!} + ... + \frac{n!}{(n-1)!1!} \end{aligned} \tag{21}$$

For instance, in the case of the *i²Snake* robot ($n = 8, m = 6$) we have $28 \leq Nc \leq 36$. This number of combinations is suitable for real time applications and can be used more generally for robotic systems having a small degree of redundancy.

D. Sparse Iterative

In [17], Fuchs suggested an iterative algorithm to solve the optimization problem:

$$\text{Find } \dot{\xi}^* \in \min_{\dot{\xi}} \quad \|J\dot{\xi} - \eta e\|_2^2 + \lambda \|\dot{\xi}\|_1 \tag{22}$$

The solution is obtained from the following recursive equation:

$$\begin{aligned} & \dot{\xi}_{k+1} = |X_k| J^T (J |X_k| J^T + \lambda \mathbb{I}) \eta e \\ & \text{with } |X_k| = diag(|\dot{\xi}_k|) \end{aligned} \tag{23}$$

λ is a real parameter that must be inside the interval $\left]0, \|J^T \eta e\|_\infty\right]$ in order to obtain a sparse solution. As the interval is changing at each sample time because of J, a normalized parameter denoted $\lambda_n \in \left[0, 1\right]$ is used. Convergence of this algorithm has been demonstrated. The stop condition is defined by $\|\dot{\xi}_{k+1} - \dot{\xi}_k\| < \epsilon_C$. (23) has a singularity when $\dot{\xi}_k = 0$. This singularity has to be taken into account at the initialization, and it can also impact the behavior of the solution (indeed small dead zone can appear when the solution is traversing the zero).

E. Sparse Linear Programming

The problem of inverse differential kinematics can be posed as a linear program as suggested in [9], which also provides sparse solutions as long as an appropriate solver is used. The problem can be expressed as follows:

$$\begin{aligned} \text{Find } \dot{\xi}^* \in \min_{\dot{\xi}} \quad & \|J\dot{\xi} - \eta e\|_1 \\ \text{subject to} \quad & W(\xi)\dot{\xi} \leq w(\xi) \end{aligned} \tag{24}$$

where $W(\xi)\dot{\xi} \leq w(\xi)$ defines a set of constraints on $\dot{\xi}$. For example, joint limitations can be taken into account as:

$$\frac{\xi_{min} - \xi}{\delta t} \leq \dot{\xi} \leq \frac{\xi_{max} - \xi}{\delta t} \tag{25}$$

The formulation in (24) by itself does not guarantee parsimony. However, since it can be transformed into a linear program, the well-known Simplex algorithm can be used to solve it and then, thanks to the intrinsic properties of that algorithm, parsimony appears whenever possible. Then, (24) can be transformed into a standard form LP problem by decomposing $\dot{\xi} = \dot{\xi}_P - \dot{\xi}_N$ and by introducing the slack variables y, z_A, z_B and z_C. One can recast (24) as:

$$\begin{aligned} & \text{Find } g^* \in \min_{g} \quad Ag + \eta 1^T e \\ & \text{subject to} \\ & \begin{pmatrix} J & -J & -\mathbb{I} & \mathbb{I} & 0 & 0 \\ W & -W & 0 & 0 & \mathbb{I} & 0 \\ 1^T & 1^T & 0 & 0 & 0 & 1 \end{pmatrix} g = \begin{pmatrix} \eta e \\ w \\ \beta(e) \end{pmatrix} \\ & g \geq 0 \end{aligned} \tag{26}$$

with $A = \begin{bmatrix} -1^T J & 1^T J & 2^T & 0^T & 0^T & 0 \end{bmatrix}$. The elements 0^T, 1^T and 2^T are row vectors composed of the scalar values 0, 1, 2, respectively. g is defined as $g = [\dot{\xi}_P^T \; \dot{\xi}_N^T \; y^T \; z_A^T \; z_B^T \; z_C]$. The constraint $1^T \dot{\xi}_P + 1^T \dot{\xi}_N = \beta(e)$ allows to guarantee that $\|\dot{\xi}\|_1 \leq \beta(e)$ and consequently if $e = 0$ then $\dot{\xi} = 0$. Typically the function $\beta(e) = \beta_0 \|e\|_1$ is used for $\beta(e)$, with β_0 a user defined parameter.

F. Sparse Hierarchical Linear Programming

The choice of the function $\beta(e)$ to guarantee the constraint on $\dot{q}$ can be critical for a large trajectory because it requires to tune the parameter β_0 and also the norm of the function. To avoid this drawback, another algorithm to solve the LP problem (24) has been developed . This algorithm is based on a hierarchical LP [18] that solves (24) through the two following steps:

$$\begin{aligned} \text{Find } \xi_1^* \in \min_{\dot{\xi}} \quad & \|J\dot{\xi} - \eta e\|_1 \\ \text{subject to} \quad & W(\xi)\dot{\xi} \leq w(\xi) \end{aligned} \tag{27}$$

$$\begin{aligned} \text{Find } \xi_2^* \in \min_{\dot{\xi}} \quad & \|\xi\|_1 \\ \text{subject to} \quad & \\ & \|J\dot{\xi} - \eta e\|_1 \leq \|J\dot{\xi}_1^* - \eta e\|_1 \\ & W(\xi)\dot{\xi} \leq w(\xi) \end{aligned}$$

The solution generated by (27) is intrinsically sparse. However the algorithm requires to solve successively two LP problems.

G. Master Manipulator

The master interface used to command the robotic system consists of a hand-held device. This device mimics the shape of a gripper, and is composed of two 6-DOF electromagnetic markers. The electromagnetic tracker used is an Ascension trakSTAR (Ascension Technology, NDI, USA). These allow the user to fully specify a desired position, orientation, and gripping angle with each hand. The mapping used between the master and slave devices is detailed below.

Let A be the base frame of the master device and B the base frame of the slave device. ${}^{A}T_{B}$ is the known homogeneous transformation from A to B with ${}^{A}P$ and ${}^{B}P$ the coordinates of a point P expressed in the frames A and B respectively. Let T_{α_t} be the homogeneous transformation from the master base frame A to the master end effector frame at time t, and T_{β_t} the homogeneous transformation from the slave base frame B to the slave end effector frame at time t. Given initial starting transformations T_{α_0} and T_{β_0}, the desired mapping is one that computes T_{β_t} such that the rotations and translations of the master end effector generate the same relative motion at the slave end effector. With regards to the rotational component R_{β_t} of T_{β_t}, this behaviour describes a system where both end effectors are rigidly linked, with an offset rotation ${}^{\alpha}R_{\beta}$. From the initial transformations ${}^{\alpha}R_{\beta} = R_{\alpha_0}^{-1}.{}^{A}R_{B}.R_{\beta_0}$, where the symbol R refers to the rotational component of the corresponding transformations. As this rigid link behaviour of the rotational components is maintained for any time t, it follows that:

$$R_{\beta_t} \;=\; {}^{A}R_{B}^{-1}\, R_{\alpha_t}\, R_{\alpha_0}^{-1}\, {}^{A}R_{B}\, R_{\beta_0}. \tag{28}$$

The desired behavior from the perspective of the translation does not follow the behavior of a rigid link, since an in-place rotation of the master device should not result in a translation of the slave device. Instead, we simply transmit the master device translation to the slave device, relative to their initial positions:

$$V_{\beta_t} \;=\; {}^{A}R_{B}\,(V_{\alpha_t} - V_{\alpha_0}) + V_{\beta_0}, \tag{29}$$

where the symbol V refers to the translational component of the corresponding transformations. The resulting relative position and rotation increment are as follow:

$$\Delta_{R_{\beta_t}} \;=\; R_{\beta_0}^{-1}\, R_{\beta_t} \;=\; {}^{A}R_{B}^{-1}\, R_{\alpha_t}\, R_{\alpha_0}^{-1}\, {}^{A}R_{B}\, R_{\beta_0} \tag{30}$$

$$\Delta_{V_{\beta_t}} \;=\; V_{\beta_t} - V_{\beta_0} \;=\; {}^{A}R_{B}\,(V_{\alpha_t} - V_{\alpha_0})\,. \tag{31}$$

Motion scaling is performed in position only, so as to preserve the mapping between the user's hand and the orientation of the robot's head. The scaling is done by multiplying the translational component by a software adjustable scaling factor set to 0.5 in the experiments.

Finally, clutching mechanics are implemented by adding a variable offset within T_{α_0}. This offset is composed of the difference between the current master end effector transformation and a saved master end effector transformation, the latter only being updated while the system is not active.

IV. Results

A. ENT Simulator

A simulator with a virtual environment was designed to perform the evaluation of each of the presented methods. The simulated tasks consist of progressively reaching point of interest of the mouth and throat in the following specific order: 1) Left then right tonsils, 2) Simulated throat tumor, 3) Simulated tongue tumor, 4) Simulated vocal cords tumor. During the study, the points of interest were highlighted in red one by one to progressively guide the user toward deep seated lesions at the bottom of the throat as depicted in Fig. 3. The simulator was developed in C++ using VTK [19]. The 3D meshes of the head are files from the BodyParts3D online library [20]. Fig. 3 shows a rendering of the simulator where the robot navigates inside the virtual head. During the user study, the participants were asked to sit in front of the simulator and to use the master interface to control the virtual robot as shown in Fig. 1 and to perform the same task with each method. The order for each method was randomized to limit any learning effect confounder. 10 users participated in the experiment. For each participant, the following information from the system was recorded at 50 Hz:

- Time stamp in seconds (1x1)
- Master transformation matrix (4x4)
- Snake tip transformation matrix (4x4)
- Joint vector (8x1)
- Target mesh ID (1x1)
- Target mesh reached boolean (1x1)
- Clutch pedal pressed (1x1)
- Clutch counter (1x1)

All the data was stacked in a CSV file and one file was generated for each user and for each method.

B. Evaluation Metrics

The evaluation of each method was performed using standard [21] and specific metrics for the proposed snake robot. The focus of this study is to find the method with the best performance based on the following three key features: intuitiveness, speed and small footprint. The intuitiveness of a system relates to the ease of use of that system for its intended purpose. In the case of robotic teleoperation, intuitive control

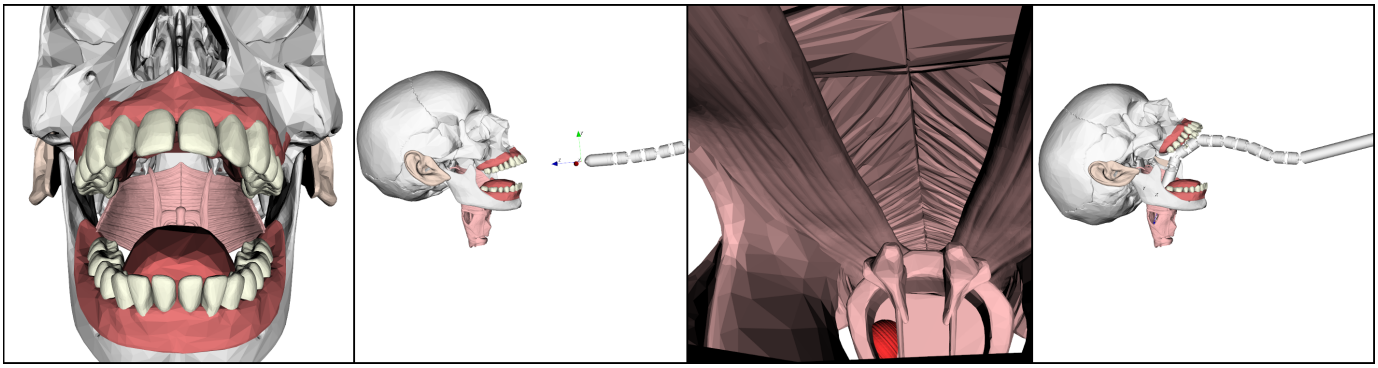

Fig. 3: Two screen-shots of the simulator. The simulator shows two separate views to the user. The left sub-view corresponds to the embedded snake camera view, while the right sub-view corresponds to the operating room outside view. The user navigates the snake-like robot using a 6 DOF master interface toward region of interest or tumors highlighted in red.

can be partially quantified by using metrics that reflect that the task was easier to perform. The total time to complete the task, the amount of clutching and the master path are all metrics that can quantify the task performance from the user point of view, and can therefore be used to evaluate how "intuitive" the control method was. In this context, the following metrics were calculated:

1) Clutching frequency: The clutching metrics reflects the necessity to re-adjust the master frequently. More clutching can reveal a poor mapping as the user would frequently press the pedal to compensate for the positioning error.

2) Robot tip distance to virtual target: This metrics looks at the distance from the tip (embedded camera view) to the virtual point of interest being targeted.

3) Rolling-joint traveled distance: The rolling-joint vector composed of ξ_3 to ξ_8 is extracted from the joint vector ξ to study the motion of only the rolling-joints. The norm of this 6D vector is calculated between two consecutive time steps and summed over the entire trajectory. This metric is used to evaluate the motion of the proximal, middle and distal sections and reflects if the robot is moving back and forth around the same position. This can be useful when a user tries to reach a point while doing small position and orientation adjustments. As some methods are easier to control than others, the joint traveled distance allows to differentiate which method is easier to control. Smaller values will reflect a method that is easier to control and are therefore preferred.

4) Robot tip path length: This metric measures the overall distance traveled by the robot tip.

5) Total time to complete the task: This metric reflects the overall difficulty of the task. Shorter time are indicative of a simpler to use method.

6) Master path length: This metric measures the overall distance traveled by the master interface and hence by the user. A larger distance implies more user fatigue and should be avoided.

7) Visited voxels volume: This metric evaluates the volume visited by each joint of the snake robot's body. The overall volume was discretized into 5 mm cubic voxels. In the case of minimally invasive surgery, the smaller the volume visited, the less impact the surgery will ultimately have on the patient and therefore, smaller values are preferred.

8) Real time performance: This metrics uses a recorded reference trajectory to assess the real-time performance of each method. The trajectory was recorded during a telemanipulation task at 50 Hz. The reference trajectory points are fed to the inverse kinematic solver of each method at a set frequency. The solver solution is then saved and the resulting cumulative error is calculated for each method. The frequency is then increased from 1 Hz ($1/50$x recorded speed) to 1500 Hz ($30\times$ recorded speed) by steps of 100 Hz. The process is then repeated for each new frequency value. This metrics highlights the limitations for real-time control. As the request frequency increases, the solver has less time to converge to a good solution. Smaller error at high frequency are desired and will result in less lag in the control.

9) Joint limit hits: This metrics counts how many joint limits are reached at each time step and outputs the cumulative joint limit hits. The higher the value, the more the amount of joints at their limit and for longer periods of time. Hitting a joint limit results in a loss of DOF and therefore this value should be as small as possible.

10) Positioning accuracy: This metric looks at the positioning error of each method to follow a desired pre-recorded user's trajectory.

C. Results

The results of the user study are presented in this section and summarized in Table III. For compactness, each evaluated method was given a short abbreviated name as follow:

- Jacobian pseudo-inverse: STD
- Linear programming: LP
- Hierarchical linear programming: HLP
- Joint limit based Jacobian: JLJ
- Sparse pseudo-L_0 norm: SPK
- Sparse iterative: SPIT

1) Clutching frequency: All methods performed similarly. SPK has a slightly higher median value. Measurement unit: count.

2) Robot tip distance to virtual target: There are no significant differences between the methods. All of them performed equally. This result confirms that all methods are working properly and converge at interactive rates making each of them

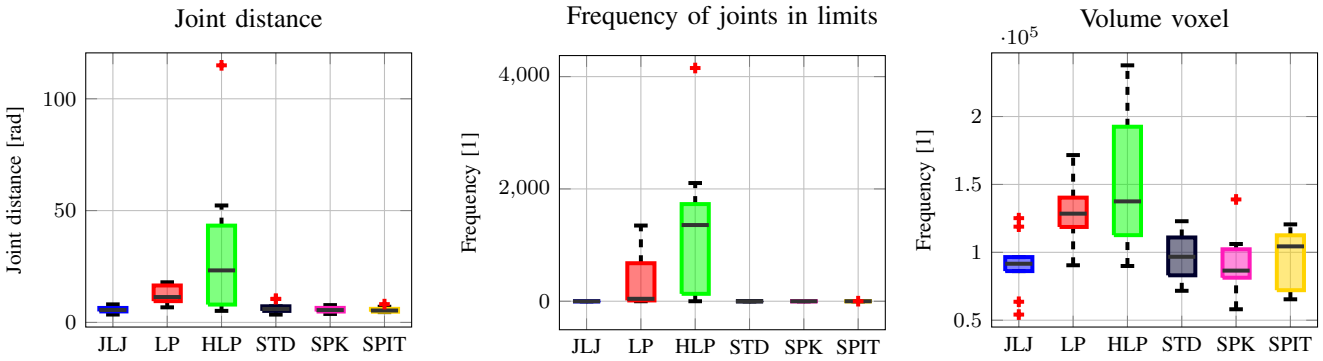

Fig. 4: Boxplots of the rolling-joint traveled distance, joint limit hits, and visited voxel volume metrics from the user study performed on the virtual ENT simulator.

suitable for telemanipulation applications. Measurement unit: meter.

3) Rolling-joint traveled distance: A Wilcoxon test was performed to compare the p-values between each method, where $p < 0.05$ is considered statistically significant. The results show that LP and HLP are significantly different to all the other methods (LP: $p = 0.002$, $p = 0.0488$, $p = 0.002$, $p = 0.002$, $p = 0.002$ for JLJ, HLP, STD, SPK and SPIT respectively; and HLP: $p = 0.0039$, $p = 0.002$, $p = 0.002$, $p = 0.002$ for JLJ, STD, SPK and SPIT respectively). There is no significant difference between the other methods. HLP followed by LP tend to use more range of the rolling joint. This result could be expected as both method try to use less joints, leading ultimately to larger joint motion, see Fig. 4. Measurement unit: radian.

4) Robot tip path length: All methods perform equally, with HLP and LP having a slightly larger distribution across the participants. During the user study, HLP and LP tended to have curlier shape than the other methods, which could sometime disrupt the user who would try to compensate for this behavior leading ultimately to larger traveled distance. Measurement unit: meter.

5) Total time to complete the task: There is no significant difference in the time required to complete the task among the participants. The standard Jacobian approach is slightly higher than the other methods with a larger distribution. Measurement unit: second.

6) Master path length: Similarly to the robot tip path length, HLP and LP tended to have curlier shape than the other methods, which could sometime disrupt the user who would try to compensate for this behavior leading ultimately to larger traveled distance. Measurement unit: meter.

TABLE III: Methods comparison (median values)

Metric	JLJ	LP	HLP	STD	SPK	SPIT
Clutching	6.5	8	7	8	7.5	7
Tip to target	5.6	5.2	5.9	5.9	6.3	5.9
Joint dist.	5.6	11.3	23.3	6.0	5.5	5.3
Tip path	1.0	1.2	1.1	1.1	1.1	1.0
Total time	51.5	54.7	57.2	52.8	48.8	46.5
Master path	2.5	2.7	2.7	2.4	2.5	2.5
Voxels (in T)	90	119	131	82	81	103
Joint limit	0	43.5	1357	0	0	0

7) Visited voxels: The visited voxels results are represented in the Fig. 4. There is a significant difference between JLJ and both LP and HLP ($p = 0.0098$ and $p = 0.0059$ respectively). LP and HLP are both significantly different from STD, SPK and SPIT (LP: $p = 0.002$, $p = 0.0059$ and $p = 0.0059$ respectively; HLP: $p = 0.002$, $p = 0.0137$ and $p = 0.0098$ respectively). There is no significant difference between the other methods. The method with the smallest median value is the SPK. SPK had the smallest footprint during the user study, and an example of such volume is depicted in Fig. 5 on the right. HLP, represented in Fig. 4 in the middle did not perform as well, as the workspace occupied to perform the same task was much larger. JLJ had an overall good performance with the smallest distribution over all the participants. JLJ is shown in Fig. 5 on the left. Measurement unit: count, voxel volume 125 mm^3.

8) Real-time performance: The results are plotted in Fig. 6. For the user study, the teleoperation frequency was set to 50 Hz to ensure a reasonable position error across all the methods. STD, JLJ and LP are capable to run at a 12× faster playback speed (600 Hz) with sub-mm accuracy.

9) Joint limits: As the robot was designed for ENT surgery, it is expected that it should be able to reach all the desired points without reaching any joint limits. Hitting a joint limit results in a loss of DOF and therefore should be avoided. During the user study, the linear programming approaches LP

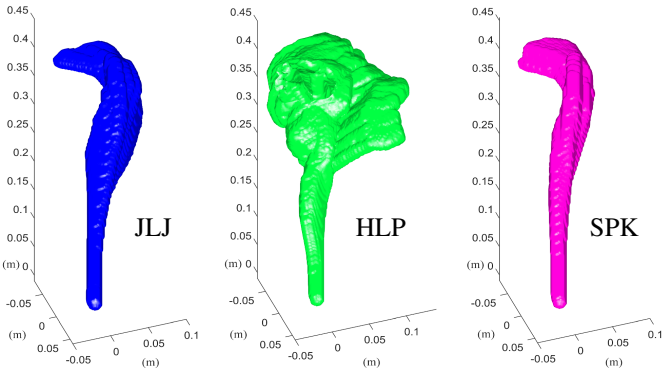

Fig. 5: Examples (from representative users) of resulting visited voxels volume rendering for the Jacobian with joint limit approach (left), hierarchical linear programming approach (middle), and the sparse pseudo-L_0 approach (right).

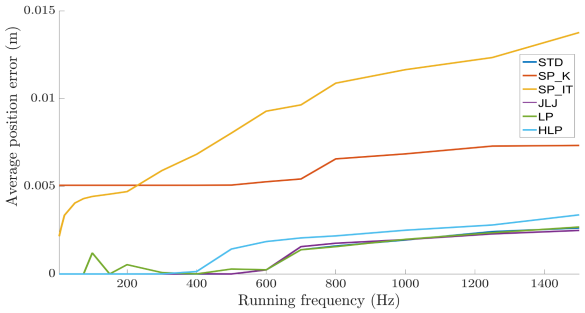

Fig. 6: Real-time performance. As the teleoperation communication frequency is increased, the resulting position accuracy decreases.

and HLP both reached joint limits multiple time but could still complete the task. STD did not reach joint limits during the study. However, hitting a joint limits with STD would most likely result in hardware damage, and should therefore be avoided. To prevent such consequences, the control algorithm stops the entire robot until a new target point can be reached. This type of implementation is common and can be found on Kuka robots (Kuka, Germany) or the dVRK [22]. However such solution is problematic during teleoperation as it interrupts the control and results in uncontrolled robot behavior. JLJ has the advantage over STD that is avoids being locked in place.

V. CONCLUSIONS

We have presented in this paper a new snake-like robot architecture for ENT surgery. The robot model was thoroughly investigated and derived, so it can be used with existing inverse kinematics approaches. Six different control methods were then implemented to be tested on a custom simulator. The methods evaluated include: Jacobian pseudo-inverse, Joint limit based Jacobian modification, sparse pseudo-L_0, sparse iterative, sparse linear programming, and sparse hierarchical linear programming. Each method was evaluated during a user study. Based on the metrics selected, the sparse pseudo-L_0 and the joint limit based Jacobian method performed better in terms of real-time capabilities, intuitiveness and small visited volume. However, the proposed methods are still not sufficient for snake-like robot teleoperation during ENT surgery. In fact, surgical teleoperation requires ensuring that there are limited collisions possible with the surrounding organs, which none of the presented method could perform. Future work will investigate hybrid approaches trying to combine all the critical features required for successful snake-robot teleoperation: such as real-time capabilities, obstacle avoidance, minimal footprint, accuracy, and economy of motion for the user.

ACKNOWLEDGMENT

The author would like to thank the user study participants, and Mohamed E. M. K. Abdelaziz for his input on VTK.

REFERENCES

[1] V. Vitiello, S. L. Lee, T. P. Cundy, and G. Z. Yang, "Emerging robotic platforms for minimally invasive surgery," *IEEE Rev Biomed Eng*, vol. 6, pp. 111–26, 2013.

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

[3] L. Zorn, F. Nageotte, P. Zanne, A. Legner, B. Dallemagne, J. Marescaux, and M. d. Mathelin, "A novel telemanipulated robotic assistant for surgical endoscopy: Preclinical application to esd," *IEEE Trans. on Biomed. Engine.*, vol. PP, no. 99, pp. 1–1, 2017.

[4] A. Degani, H. Choset, A. Wolf, and M. Zenati, "Highly articulated robotic probe for minimally invasive surgery," in *IEEE Int. Conf. on Robot. and Autom.*, 2006, pp. 4167–4172.

[5] H. Choset and W. Henning, "A follow-the-leader approach to serpentine robot motion planning," *Journal of Aerospace Engineering*, vol. 12, no. 2, pp. 65–73, 1999.

[6] 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 *IEEE Int. Conf. on Robot. and Autom.*, 2015, pp. 2361–2367.

[7] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *Jou. of Intel. Rob. and Sys.*, vol. 3, no. 3, pp. 201–212, 1990.

[8] F. Flacco, A. De Luca, and O. Khatib, "Control of redundant robots under hard joint constraints: Saturation in the null space," *IEEE Trans on Robot.*, vol. 31, no. 3, pp. 637–654, 2015.

[9] V. M. Gonçalves, P. Fraisse, A. Crosnier, and B. V. Adorno, "Parsimonious kinematic control of highly redundant robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 65–72, 2016.

[10] J. W. Suh, K. Y. Kim, J. W. Jeong, and J. J. Lee, "Design considerations for a hyper-redundant pulleyless rolling joint with elastic fixtures," *IEEE-Asme Trans. on Mechat.*, vol. 20, no. 6, pp. 2841–2852, 2015.

[11] A. Jeanneau, J. Herder, T. Laliberte, and C. Gosselin, "A compliant rolling contact joint and its application in a 3-dof planar parallel mechanism with kinematic analysis," in *2004 ASME Int. Des. Engine./Comp. and Info. Techn. Conf.* American Society of Mechanical Engineers, 2004, pp. 689–698.

[12] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Trans. on man-machine syst.*, vol. 10, no. 2, pp. 47–53, 1969.

[13] K. Leibrandt, P. Wisanuvej, G. Gras, J. Shang, C. A. Seneci, P. Giataganas, V. Vitiello, A. Darzi, and G. Z. Yang, "Effective manipulation in confined spaces of highly articulated robotic instruments for single access surgery," *IEEE Rob. and Auto. Lett.*, vol. 2, no. 3, pp. 1704–1711, 2017.

[14] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[15] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

[16] S. Muthukrishnan, "Data streams: Algorithms and applications," *Found. and Tren. in Theo. Comp. Scien.*, vol. 1, no. 2, pp. 117–236, 2005.

[17] J. J. Fuchs, "Convergence of a sparse representations algorithm applicable to real or complex data," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 598–605, 2007.

[18] H. Isermann, "Linear lexicographic optimization," *Operations-Research-Spektrum*, vol. 4, no. 4, pp. 223–228, 1982.

[19] W. J. Schroeder, B. Lorensen, and K. Martin, *The visualization toolkit: an object-oriented approach to 3D graphics*. Kitware, 2004.

[20] "Bodyparts3d, the database center for life science licensed under cc attribution-share alike 2.1 japan."

[21] A. Steinfeld, T. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz, and M. Goodrich, "Common metrics for human-robot interaction," in *SIGCHI/SIGART conf. on Human-robot inter.* ACM, 2006, pp. 33–40.

[22] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. DiMaio, "An open-source research kit for the da vinci surgical system," in *IEEE Intl. Conf. on Robot. and Auto.*, Hong Kong, China, 2014, pp. 6434–6439.