

Optimization of Surgical Robotic Instrument Mounting in a Macro–Micro Manipulator Setup for Improving Task Execution

Francesco Cursi¹, Student Member, IEEE, Weibang Bai², Member, IEEE, Eric M. Yeatman³, Fellow, IEEE, and Petar Kormushev⁴, Member, IEEE

Abstract—In minimally invasive robotic surgery, the surgical instrument is usually inserted inside the patient’s body through a small incision, which acts as a remote center of motion (RCM). Serial-link manipulators can be used as macro robots on which microsurgical robotic instruments are mounted to increase the number of degrees of freedom of the system and ensure safe task and RCM motion execution. However, the surgical instrument needs to be placed in an appropriate configuration when completing the motion tasks. The contribution of this article is to present a novel framework that preoperatively identifies the best base configuration, in terms of Roll, Pitch, and Yaw angles, of the microsurgical instrument with respect to the macro serial-link manipulator’s end effector in order to achieve the maximum accuracy and dexterity in performing specified tasks. The framework relies on hierarchical quadratic programming for the control, genetic algorithm for the optimization, and on a resilience to error strategy to make sure deviations from the optimum do not affect the system’s performance. Simulation results show that the mounting configuration of the surgical instrument significantly impacts the performance of the whole macro–micro manipulator in executing the desired motion tasks, and both the simulation and experimental results demonstrate that the proposed optimization method improves the overall performance.

Index Terms—Genetic algorithm (GA), minimally invasive robotic surgery (MIRS), redundant robots, robotic configuration optimization.

I. INTRODUCTION

MINIMALLY invasive robotic surgery (MIRS) has been introduced in surgical operations due to the need to improve precision and augment the surgeon’s capabilities, and therefore, reduce the patient’s trauma.

In robotically assisted surgical procedures, the surgical instrument is inserted inside the patient’s body through small incisions. This, however, restricts the motion of the robot, which is not allowed to move tangentially to the hole plane, but can only pivot and translate in the insertion direction about the insertion point. One of the challenges for the control of surgical robots is guaranteeing safe execution of the constrained motion at the insertion point, also known as remote center of motion (RCM). This constraint leads the robotic system to lose two degrees of freedom (DOFs) [1].

Different approaches have been developed to overcome the issue of loss of mobility due to the RCM motion, and they can typically be divided into design-based and control-based strategies [2]. Design-based strategies focus on constraining the RCM mechanically, by designing robotic structures, like parallel mechanisms [3], [4], to guarantee RCM. The drawback of these devices is that, since the mechanical RCM is at a fixed location with respect to the manipulator, they are usually specific for a certain type of operation and have complex structures [5], [6].

Control-based techniques, instead, allow RCM motion by properly tuning the control algorithm. These approaches simply require the kinematic model of the robot and can be employed in different situations to ensure that the instrument is directed through the RCM at all times. In order to achieve that, the robotic system must have enough DOFs. Therefore, a common approach is to use traditional serial-link manipulators on which the microsurgical instrument is mounted [7], [8] by choosing a suitable configuration for the base of the surgical instrument with respect to the macro-robot’s end effector. Various approaches have been employed to control surgical robots under RCM constraint, such as the augmented Jacobian [1], [9], [10], projection in the null space method [11], and multiobjective optimization with constraints [12], which exploit the system redundancy by considering different subtasks.

Manuscript received 21 November 2021; accepted 23 April 2022. Date of publication 17 May 2022; date of current version 4 October 2022. This work was supported by Engineering and Physical Sciences Research Council (EPSRC) programme grant Micro-Robotics for Surgery under Grant EP/P012779/1. This article was recommended for publication by Associate Editor C. Yang and Editor A. Menciassi upon evaluation of the reviewers’ comments. (Corresponding author: Weibang Bai.)

Francesco Cursi is with Hamlyn Centre and Robot Intelligence Laboratory, Imperial College London, SW72AZ London, U.K. (e-mail: cursifrancesco@gmail.com).

Weibang Bai is with Hamlyn Centre and the Department of Computing, Imperial College London, SW72AZ London, U.K. (e-mail: wbbai@imperial.ac.uk).

Eric M. Yeatman is with Hamlyn Centre and the Department of Electrical and Electronic Engineering, Imperial College London, SW72AZ London, U.K. (e-mail: e.yeatman@imperial.ac.uk).

Petar Kormushev is with the Robot Intelligence Lab, Imperial College London, London SW72AZ, U.K. (e-mail: p.kormushev@imperial.ac.uk).

This article has supplementary downloadable material available at <http://ieeexplore.ieee.org>, provided by the authors. The video summarizes the submitted work “Optimization of Surgical Robotic Instrument Mounting in a Macro–Micro Manipulator Setup for Improving Task Execution.” It starts with a brief summary of the method and then shows results both for the simulation and real world experiments. This video is known to be compatible with VLC Media player, version 3.08.

This article has supplementary material provided by the authors and color versions of one or more figures available at <https://doi.org/10.1109/TRO.2022.3171097>.

Digital Object Identifier 10.1109/TRO.2022.3171097

The main drawbacks of these approaches are that they either may not always guarantee that the different subtasks are always satisfied, e.g., due to conflicts such as in the augmented Jacobian method [13], or do not allow to include additional constraints. Moreover, they only target surgical robots with few DOFs. In addition, when serial-link articulated robots are employed, the surgical instrument needs to be attached in a proper configuration in order to guarantee correct and safe task execution. Both in teleoperation and in autonomous control, the whole robotic structure needs to accurately fulfil the commanded motion task, satisfying the RCM and additional constraints, but to also allow as much dexterity as possible.

Different works have focused on finding the optimal placement of a robotic system, in order to guarantee a good performance in solving a given motion task. This problem has been of great interest especially in the manufacturing industry. In [14], the authors look for optimal robot-workpiece placement of a redundant manufacturing robot for a welding process through particle swarm optimization. Their approach, however, yielded an optimal configuration residing within a small region. This may lead to a deterioration of the performance if the placement is not accurate. Spensieri *et al.* [15] used the Nelder-Mead method for finding the optimal base placement of a manufacturing robot. Yet, the converge properties have not been proved, except for small problems [16].

Usually robots in manufacturing are required to complete the same single specified task over time, therefore, finding the optimal robot base placement may suffice, since it is required only once. In robotic surgery, instead, the task may vary depending on the surgical procedures to complete, and thus, the placement may need to change accordingly. In the field of robotic surgery, a few researches have focused on the optimal placement of a robot. For example, in [17], the authors focused on seeking optimal base positioning of a robot with an ultrasound probe for radiotherapy. However, their work did not require any consideration about RCM constraint. Konietzschke *et al.* [18] utilized the genetic algorithm (GA) to optimally design a robot manipulator to perform a surgical task; the main drawback is that different tasks may require different new designs.

Other works, instead of optimizing for robot design or placement, concentrated on the optimal insertion point position. In [19], the authors used CT images to manually identify the set of possible port locations, and then, applied a search space method among the possible port locations to find the best one. In this work, however, the RCM motion was achieved thanks to the robot's mechanical design. In [2], instead, the robotic setup consists of a serial-link manipulator and a surgical instrument. This work is concerned with the problem of choosing a location for the RCM relative to the manipulator in order to maximize the system's performance. Their optimization only consisted of two position parameters along x and z and used a grid of 121×121 points as a search space.

The limitation of focusing on the optimal placement of the base of the robot is that it requires a large effort to move the whole robot around. This might also require a redesign of the whole operating room for each surgical procedure. Optimizing the robot design [20] on the other hand, would probably require

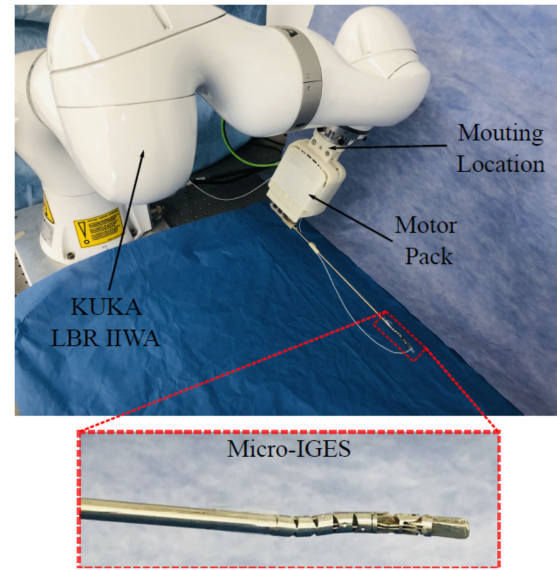


Fig. 1. Macro-micro manipulator setup consisting of the KUKA Macro serial-link manipulator and the Micro-IGES as a microsurgical robotic instrument.

a new design when a new task is assigned. The works focusing on the placement of the insertion point, instead, were mostly employing robots like the DaVinci, where the RCM motion is obtained mechanically. Moreover, the optimization of the placement of the insertion point should be able to include information provided by the surgeons, who take their decisions according to preoperative surgical requirements and patient's anatomy [21]. In addition, most of these works took into account only the system dexterity, without consideration of motion accuracy or resilience to placement errors. Configurations with high dexterity are not guaranteed to achieve good tip tracking performance, and viceversa, configurations leading to small tracking errors may not lead to high dexterity. Because of the RCM constraint, also the accuracy in tracking the RCM motion needs to be taken into account. Furthermore, the optimal configurations found do not take into account placement errors, and therefore, may reside in regions where small deviations from the optimum lead to large deterioration of the performance.

This manuscript is thus intended to provide a novel framework to identify the optimal mounting setup, by choosing the best base configuration of the microsurgical robotic instrument with respect to its macro serial-link manipulator's end-effector frame, to ensure the maximum accuracy and dexterity in performing specified tasks. The surgical robotic instrument's base configuration is here described by means of Roll, Pitch, and Yaw (RPY) angles, which are chosen as optimization variables. Optimizing only for the connection between the two robots makes the whole macro-micro manipulator more flexible and easily adaptable to different surgical procedures with minimum effort. This work focuses on the use of a KUKA robotic arm as a macrorobot, and the Micro-IGES surgical robotic instrument [22] (see Fig. 1) as a microrobot, to simulate a tumor resection task. However, the framework can be generalized to any other robotic structures and tasks.

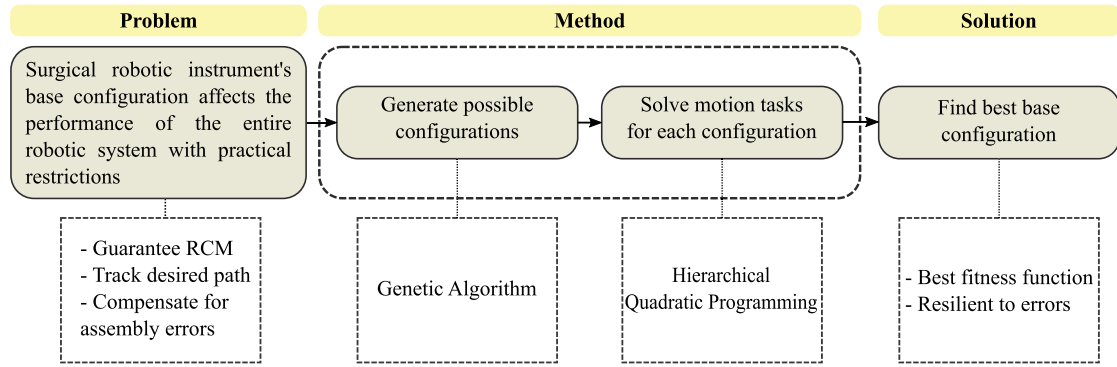


Fig. 2. Roadmap of the proposed framework for finding the optimal base configuration of the surgical robotic instrument, with respect to the serial-link manipulator's end-effector frame.

To the best of the authors' knowledge, there is no other published work analyzing the effects of the connection between a surgical robotic instrument and a serial-link manipulator, and focusing on finding the optimal connection in order to improve the task execution. To address this problem, we propose a novel framework consisting of an optimization algorithm, an optimal control strategy, and a resilience to error approach to limit the loss of performance due to possible assembly errors.

The optimization algorithm searches through all possible base configurations of the surgical robotic instrument and finds the optimal one, while the optimal control strategy is used to ensure proper task execution and acts as a testbench to compare the different configurations. Fig. 2 shows the roadmap of the proposed framework.

For the control problem, the hierarchical quadratic programming (HQP) [23] approach is adopted, since it allows to optimally solve multiple prioritized subtasks while satisfying additional constraints. With regards to the optimization of the surgical robotic instrument's base configuration, the GA is used. The choice of an heuristic global optimization approach like GA over nonheuristic approaches is dictated by the fact that nonheuristic approaches are generally based on some knowledge about the robotic structure and are more model-specific, thus not as generalizable as the heuristic ones. Among other heuristic approaches such as particle swarm optimization or Bayesian optimization, GA was chosen thanks to its ability to explore large search spaces, more likely find global minima, and to its capability to better deal with discrete variables. An appropriate fitness function is introduced, presenting a novel approach for computing the robot dexterity measure when multiple prioritized subtasks are assigned. However, the optimal configuration identified by the solver may reside in a region where small deviations from the optimal may lead to much worse performance. Consequently, a strategy to find an optimal solution in a neighborhood that guarantees a good performance is also introduced.

The main contributions of this work are as follows:

- 1) Analyzing how the mounting configuration between a macro serial-link robot and a microsurgical robotic instrument affects the performance of the whole system in performing a specified task.

- 2) Proposing a framework to identify the optimal mounting configuration in order to maximize the accuracy and dexterity in performing a desired task.
- 3) Proposing a fitness function and a proper resilience to error strategy to be included in the framework in order to find the optimal solution and guarantee that deviations from the optimum do not deteriorate the system's performance in fulfilling the desired task.

Our simulation results show that the configuration of the surgical robotic instrument with respect to the serial-link manipulator significantly impacts the performance of the whole macro-micro manipulator in executing the desired motion tasks, thus justifying the importance of finding optimal robot's configurations. In addition, our proposed strategy for increasing resilience to errors is shown to be effective in finding optimal configurations, which lie in regions where small variations in the tracking errors and dexterity occur, thus ensuring that assembly errors will not deteriorate the system's performance.

This article is structured as follows. In Section II, the Micro-IGES surgical robotic instrument is described and the kinematic model of the whole macro-micro manipulator composed by the KUKA arm and the Micro-IGES robot is derived. Section III describes state-of-the-art approaches for controlling redundant manipulators and the chosen HQP approach. Additionally, an overview of the GA for function optimization is presented. Section IV presents the proposed framework, describing the control strategy to guarantee the execution of the desired motion task, while satisfying the RCM and additional constraints, and formulating the optimal configuration problem to be solved with the GA. Results are then shown in Section V, and finally, Section VI concludes this article.

II. ROBOTIC SYSTEM DESCRIPTION

In this section, an overview of the robotic system setup is presented, describing the Micro-IGES surgical robotic instrument and the KUKA LBR IIWA robotic arm.

A. Micro-IGES Surgical Robotic Instrument

1) *Robot Description:* The Micro-IGES [22], [24] (see Fig. 3) is a surgical robotic instrument, composed of a rigid shaft (27 cm) and a flexible section (39 mm at zero configuration).

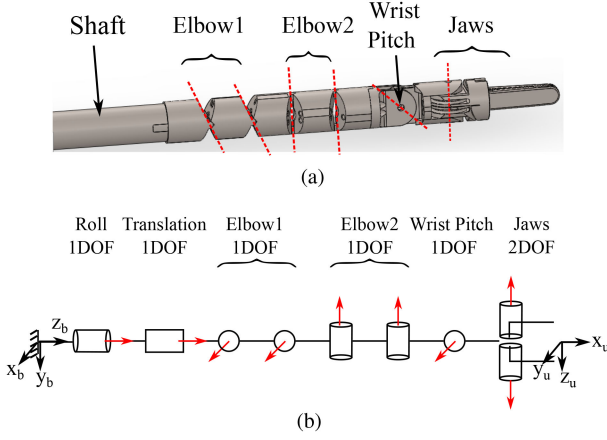


Fig. 3. Micro-IGES surgical robotic instrument. (a) CAD model. (b) Kinematic model.

The shaft is responsible for the roll and translation DOFs. The articulated end, instead, consists of two elbows for pitch and yaw, with each elbow made of a pair of coupled joints, a 1-DOF revolute joint for the wrist pitch and the jaws. The jaws provide two more DOFs: one for the wrist yaw and one for the gripper's opening/closing. Each joint of the articulated part is driven by an antagonistic pair of tendons, with each pair being connected to the corresponding driving capstan at the proximal drive unit. The coupling of the two pairs of joints of the elbows occurs at the driving unit: the two capstans that drive the two serial joints for each DOF of the elbow (pitch and yaw) are coupled by a series of gears with 1:2 ratio. Due to the current setup, in this work, only 5 DOF are considered (Roll, Elbow 1, Elbow 2, Wrist Pitch, and Wrist Yaw), since the translation is provided by the KUKA arm and gripper opening/closing is irrespective here.

2) *Robot Kinematic Model*: The nonlinearities in tendon transmission make the mathematical derivation of the system kinematics and dynamics tedious. In addition, tendon-driven systems for minimally invasive surgery usually lack sensors at the distal side, therefore, joint values cannot be measured and controlled directly. Because of the general nonlinear mapping between motors and joints $\mathbf{q}_u = \mathbf{f}(\boldsymbol{\theta}_u)$, being $\boldsymbol{\theta}_u$ the vector of motor positions and $\mathbf{q}_u$ the vector of joint positions, the kinematic model of the robot, with respect to its base frame $\{\text{RF}_b\} = [x_b, y_b, z_b]$, can be rewritten as

$$\begin{aligned} {}^bT_u &= {}^bT_q(\mathbf{q}_u) = {}^bT_u(\boldsymbol{\theta}_u) \\ {}^b\mathbf{v}_u &= {}^bJ_q(\mathbf{q}_u)\dot{\mathbf{q}}_u = {}^bJ_q(\mathbf{q}_u)L(\mathbf{q}_u)\dot{\boldsymbol{\theta}} = {}^bJ_u(\boldsymbol{\theta}_u)\dot{\boldsymbol{\theta}}_u \end{aligned} \quad (1)$$

where ${}^bT_q, {}^bT_u \in \mathbb{R}^{4 \times 4}$ describe the Cartesian end-effector pose in terms of joint and motor positions, respectively, ${}^b\mathbf{v}_u \in \mathbb{R}^6$ the Cartesian end-effector twist (linear and angular velocities vector), bJ_q is the Cartesian task Jacobian with respect to the joint variables, and bJ_u the Jacobian with respect to the motor values. The matrix L , with $L_{i,j} = \frac{\partial q_{u,i}}{\partial \theta_{u,j}}$, is the motor to joint differential matrix [25]. In this work, we employed the same motor to joint mapping as in [26]. The control problem for the Micro-IGES can, therefore, be formulated as a function

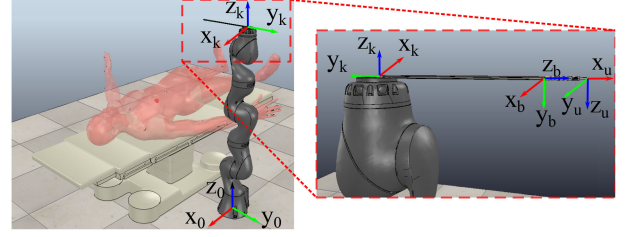


Fig. 4. Simulation environment showing the Macro-Micro manipulator.

of the motor values, which can be directly measured and controlled [27]. As described in Section II-A1, the system's state is expressed by $\boldsymbol{\theta}_u = [\theta_R \ \theta_{e1} \ \theta_{e2} \ \theta_W \ \theta_Y]$ for the Roll, Elbow 1, Elbow 2, Wrist Pitch, and Wrist Yaw motion.

B. Macro Robotic Manipulator

The KUKA LBR IIWA is an articulated industrial robot with seven joints ($\mathbf{q}_k \in \mathbb{R}^7$). Thanks to its serial-link manipulator design and direct transmission (the motors are placed directly at the joints), the kinematic model of this robot can be easily computed by means of the Denavit–Hartenberg convention. Therefore, the kinematics of the KUKA can be expressed with respect to its base reference frame $\{\text{RF}_0\} = [x_0, y_0, z_0]$ as

$$\begin{aligned} {}^0T_k &= {}^0T_k(\mathbf{q}_k) \\ {}^0\mathbf{v}_k &= {}^0J_k(\mathbf{q}_k)\dot{\mathbf{q}}_k. \end{aligned} \quad (2)$$

C. Macro-Micro Manipulator Model

The macro-micro manipulator is composed of the Micro-IGES and the KUKA robots (see Fig. 4). The Micro-IGES motor package is directly attached to the KUKA end effector. In our simulation environment, however, to simplify the visualization, the Micro-IGES is considered to be directly attached to the KUKA, just neglecting a vertical translation corresponding to the height of the Micro-IGES motor pack. In total, the system has 12 DOF (seven for the KUKA and five for the Micro-IGES). The robot state is then described by $\mathbf{q} = [\mathbf{q}_k \ \boldsymbol{\theta}_u]^T$. To compute the kinematics of the whole system, one approach could be to generate a single model of the whole manipulator and use standard modeling techniques, such as the Denavit–Hartenberg convention. In this work, however, we propose a more flexible approach that consists in combining the two models of the robots.

The whole macro-micro manipulator end-effector's pose, with respect to the KUKA base, can thus be computed as

$$T(\mathbf{q}) = {}^0T_k(\mathbf{q}_k) {}^kT_b {}^bT_u(\boldsymbol{\theta}_u) \quad (3)$$

where 0T_k is the KUKA end-effector pose, ${}^kT_b \in \mathbb{R}^{4 \times 4}$ is a fixed transformation matrix from the KUKA end-effector to the Micro-IGES base frame, and bT_u is the Micro-IGES pose with respect to its base.

Decomposing (3) into position and orientation, results in

$$P(\mathbf{q}) = {}^0P_k + {}^0R_k {}^kP_{b,k} + {}^0R_k {}^kR_b {}^bP_{u,b} \quad (4a)$$

$$R(\mathbf{q}) = {}^0R_k {}^kR_b {}^bR_u \quad (4b)$$

where 0P_k is the KUKA end-effector position in its base frame, ${}^kP_{b,k}$ is the Micro-IGES's base position with respect to the KUKA end-effector frame, ${}^bP_{u,b}$ is the Micro-IGES's end-effector position with respect to its base frame, 0R_k is the KUKA end-effector rotation matrix in its base frame, kR_b is the Micro-IGES's base rotation matrix in KUKA's end-effector frame, and bR_u the Micro-IGES end-effector rotation matrix with respect to its base frame.

By deriving (4a) with respect to time, it yields $\dot{P} = {}^0\dot{P}_k + S({}^0\omega_k)({}^0P_{b,k} + {}^0P_{u,b}) + {}^0R_b {}^b\dot{P}_{u,b}$, with $S \in \mathbb{R}^{3 \times 3}$ being the skew-symmetric matrix of cross-product and $\omega_k \in \mathbb{R}^3$ KUKA's end-effector angular velocity. Being $\omega_k = J_{o,k}\dot{q}_k$, with $J_{o,k} \in \mathbb{R}^{3 \times 7}$ the KUKA orientation Jacobian, and from the definition of the cross product, it results that

$$\begin{aligned} \dot{P} &= {}^0J_{l,k}\dot{q}_k \\ &+ \left[{}^0J_{o,1} \times {}^0P_{u,k} \quad \dots \quad {}^0J_{o,7} \times {}^0P_{u,k} \right] \dot{q}_k \\ &+ {}^0J_{l,u}\dot{\theta}_u \\ &= \begin{bmatrix} {}^0\hat{J}_{l,k} & {}^0J_{l,u} \end{bmatrix} \begin{bmatrix} \dot{q}_k \\ \dot{\theta}_u \end{bmatrix} = {}^0J_{l,tot}\dot{q} \end{aligned} \quad (5)$$

where $J_{l,k} \in \mathbb{R}^{3 \times 7}$ and $J_{l,u} \in \mathbb{R}^{3 \times 5}$ are the Jacobians of for the linear velocity of each robot, and $\hat{J}_{l,k} = J_{l,k} + [J_{o,1} \times P_{u,k} \dots J_{o,7} \times P_{u,k}]$.

With regards to the orientation, by deriving (4b), we have $\dot{R} = {}^0\dot{R}_k {}^kR_u + {}^0R_b {}^b\dot{R}_u = S({}^0\omega_k){}^0R_u + {}^0R_b S({}^b\omega_u){}^bR_u$, and then,

$$\begin{aligned} \dot{R} &= S(\omega){}^0R_u = (S({}^0\omega_k) + S({}^0\omega_u)){}^0R_u \\ \omega &= {}^0\omega_k + {}^0\omega_u = \begin{bmatrix} {}^0J_{o,k} & {}^0J_{o,u} \end{bmatrix} \begin{bmatrix} \dot{q}_k \\ \dot{\theta}_u \end{bmatrix} = {}^0J_{o,tot}\dot{q}. \end{aligned} \quad (6)$$

By combining (5) and (6), the macro-micro manipulator's end-effector twist can be computed as

$$v = \begin{bmatrix} \dot{P} \\ \omega \end{bmatrix} = \begin{bmatrix} J_{l,tot} \\ J_{o,tot} \end{bmatrix} \dot{q} = \begin{bmatrix} J_{KUKA} & J_{IGES} \end{bmatrix} \dot{q} = J_{tot}\dot{q} \quad (7)$$

with $J_{KUKA} \in \mathbb{R}^{6 \times 7}$ and $J_{IGES} \in \mathbb{R}^{6 \times 5}$ describing the contribution of each robot to the motion.

III. PROBLEM FORMULATION

In this section, a brief overview of the methods for the redundant robot control is presented, along with the description of the GA, employed for the optimization of the robot configuration.

A. Redundancy Resolution

The goal of inverse kinematics is to compute the proper joint angles $q \in \mathbb{R}^{n_j}$ and velocities $\dot{q}$ that allow the system to execute a desired motion task. The linearity at the velocity level is usually exploited to compute joint velocities from

$$\dot{h} = J(q)\dot{q} \quad (8)$$

where J is the Jacobian matrix and $\dot{h} \in \mathbb{R}^{m_p}$ is a desired task.

In case of redundant robots, the number of available joints n_J to execute the trajectory is larger than the number of constraints imposed by the desired task dimension m_p . Therefore, the solution to problem (8), if it exists, is not unique. Different techniques have been developed to solve this problem, such as the extended Jacobian technique [28], [29] or the augmented Jacobian [30]. Some additional tasks are introduced so as to solve a determined single system of linear equations.

Those approaches, however, do not allow to consider task prioritization. In fact, all tasks are given the same level of priority and this may lead to conflicting tasks. In many cases, some tasks have lower priority than others, meaning that their achievement is less important. The gradient projection method allows to take task prioritization into consideration. Considering the case of control at velocity level, the general solution of (8) is [31]

$$\dot{q} = J^\dagger \dot{x}_d + \mathcal{P} \dot{q}_{null} \quad (9)$$

where $J^\dagger$ is the pseudoinverse of J , $\mathcal{P} = I - J^\dagger J$ is the projector onto the null space of J , and $\dot{q}_{null}$ is an arbitrary vector, responsible for the self-motions of the system: the robot can move without affecting the execution of the main motion task. Low priority tasks can be executed by being projected into the null space of higher priority tasks, without affecting the higher priority tasks [32].

Nevertheless, the gradient projection method does not permit to include system constraints. Recently, especially in the field of humanoid robotics, where researchers have to deal with high level of redundancy, a common approach to solve stacks of prioritized tasks subjected to different constraints (both equality and inequality) consists in applying the HQP framework introduced by [23], where the control problem is formulated as

for $n = 1 \dots N$

$$\begin{aligned} \dot{q}_n &= \arg \min_{\dot{q}} \frac{1}{2} \|\dot{h}_n - J_{h,n}\dot{q}\|^2 + \frac{\lambda}{2} \|\dot{q}\|^2 \\ \text{s.t. } J_{h,0}\dot{q} &= J_{h,0}\dot{q}_0 \\ A_{in,0}\dot{q} &\leq b_{in,0} \\ &\vdots \\ J_{h,n-1}\dot{q} &= J_{h,n-1}\dot{q}_{n-1} \\ A_{in,n}\dot{q} &\leq b_{in,n} \end{aligned} \quad (10)$$

where N is the number of prioritized subtasks, $\dot{h}_n$ is a desired task to achieve, $A_{in,n}$, $b_{in,n}$ describe additional inequality and equality constraints for each motion task. This formulation allows to find the optimal solution $\dot{q}_n$, without violating the higher priority tasks and the imposed constraints. $A_{in,0}$, $b_{in,0}$, $J_{h,0}$ can be set to zero or empty, as only used for initialization. λ is a constant coefficient for regularization. If only equality constraints are used, the solution of (10) coincides with (9).

B. Genetic Algorithm (GA)

The GA [33] belongs to evolutionary optimization methods, which try to find global optima of a cost function by mimicking

the mechanisms of natural selection and genetic evolution. The main advantages of the GA are as follows:.

- 1) it is derivative free, thus it does not require to compute the derivatives of the cost function;
- 2) it finds the cost function optima in a trial and error way, by using multiple individuals per iteration;
- 3) it allows to have very large populations, and thus, effectively explore large search spaces.

Due to its capabilities, the GA has been widely used to identify optimal design configurations [34]. Different works employed the GA for optimal robotic design [35]–[37].

In this work, MATLAB Optimization Toolbox has been used and the following steps are performed.

- 1) *Initialization*: The algorithm begins by creating an initial population of *PopSize* individuals. This initial population can be assigned by the user or generated by means of the *CreationFunction*.
- 2) *Function Evaluation*: The fitness function for each individual in the current population is computed.
- 3) *Parents Selection*: The individuals are ranked depending on their corresponding value of the fitness function (*FitValues*). The *ScalingFunction* is used to scale *FitValues* and obtain *ScaledValues*. The *ScaledValues* are then exploited by the *SelectionFunction* to choose the appropriate parents for the next generation.
- 4) *Children Generation*: From the *Parents*, *Children* are generated in different ways. A certain number of individuals are passed directly onto the next generation (*Elite Children*). A certain percentage of the *Children* is instead obtained by means of the *Crossover Function*, and some others are randomly obtained through the *Mutation Function*.
- 5) *Migrate Population*: Finally, the obtained *Children* are used to replace the current *Population*, and thus, obtain a new population.

Steps 2–5 are repeated until the stopping criteria are met. Generally, the algorithm stops if the fitness value is lower than a specific threshold, if the maximum number of generations is reached, or if the best fitness value does not change for a certain number of generations (*Stall Generations*). At the end of the procedure, the population of the latest generation is returned.

IV. METHOD

In this section, the framework to find the best surgical robotic instrument's base configuration in order to perform a desired task is presented, describing the chosen control method and formulating the optimization problem.

A. Motion Control

The robotic system comprised of the KUKA and the Micro-IGES robots is highly redundant. In fact, the total number of DOFs is 12 here. In order to optimally exploit the hyperredundancy of the system, different prioritized subtasks can be specified and the HQP framework (10) adopted. In a surgical motion task like tumor resection, at least two prioritized subtasks

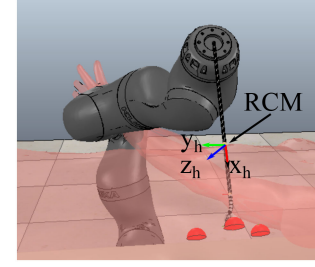


Fig. 5. RCM representation.

can be specified: keeping the RCM fixed and follow a desired path to remove the tumor.

1) *Remote Center of Motion (RCM)*: The RCM is a point, usually the surgical instrument's insertion point, where only rotations and insertion motions can be achieved. During laparoscopic procedures, for instance, the RCM coincides with the location of the hole on the patient's body where the surgical instrument is then inserted, as shown in Fig. 5. Therefore, the motion at the RCM needs to be constrained along the surface tangential directions (y and z in Fig. 5). Consequently, the primary subtask is to constrain the RCM and the corresponding cost function can be specified as $\|\dot{\mathbf{h}}_{\text{RCM}} - \mathbf{J}_{\text{RCM}}\dot{\mathbf{q}}\|^2$. $\mathbf{J}_{\text{RCM}}$ allows to compute the velocity of the RCM based on the robot kinematics.

The position of the RCM with respect to the macro-micro manipulator's base is computed similarly as in (4a) as $\mathbf{P}_{\text{RCM}} = {}^0\mathbf{P}_k + {}^0\mathbf{R}_k^T \mathbf{P}_{\text{RCM},k}$, with ${}^k\mathbf{P}_{\text{RCM},k}$ being the RCM position with respect to the KUKA's end effector, in the end-effector's frame. By deriving with respect to time, as in (5), it turns out that

$$\begin{aligned} \dot{\mathbf{P}}_{\text{RCM}} &= {}^0\mathbf{J}_{l,k}\dot{\mathbf{q}}_k \\ &+ \begin{bmatrix} {}^0\mathbf{J}_{o,1} \times {}^0\mathbf{P}_{\text{RCM},k} & \dots & {}^0\mathbf{J}_{o,7} \times {}^0\mathbf{P}_{\text{RCM},k} \end{bmatrix} \dot{\mathbf{q}}_k \\ &= \begin{bmatrix} {}^0\hat{\mathbf{J}}_{\text{RCM}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_k \\ \dot{\boldsymbol{\theta}}_u \end{bmatrix} = {}^0\mathbf{J}_{\text{RCM}}\dot{\mathbf{q}}. \end{aligned} \quad (11)$$

The aforementioned equation shows that the contribution of the surgical robotic instrument to the RCM is null, as the RCM occurs on the shaft.

To simplify the control problem, in this work, the RCM motion is expressed with respect to the insertion point

frame in Fig. 5. Therefore, $\mathbf{J}_{\text{RCM}} = \begin{bmatrix} {}^h\mathbf{R}_0^T {}^0\mathbf{J}_{l,\text{RCM}} \\ {}^h\mathbf{R}_0^T {}^0\mathbf{J}_{o,\text{RCM}} \end{bmatrix}$, with

${}^0\mathbf{R}_h = {}^h\mathbf{R}_0^T$ the rotation matrix of the hole frame with respect to the KUKA base frame, and ${}^0\mathbf{J}_{l,\text{RCM}}$ and ${}^0\mathbf{J}_{o,\text{RCM}}$ the linear and orientation part of the RCM Jacobian. Thanks to this transformation, assuming the hole fixed, the RCM motion to consider is only along the y and z components. Consequently, the desired RCM velocity is set to $\dot{\mathbf{h}}_{\text{RCM}} = [0, 0]^T$ in order to guarantee the RCM motion in the tangential directions to be fixed and $\mathbf{J}_{\text{RCM}} \in \mathbb{R}^{2 \times 12}$. In cases where the RCM is not stationary (like patient breathing), it is still possible to track the RCM by properly specifying the velocity $\dot{\mathbf{h}}_{\text{RCM}}$. In these cases,

however, the RCM velocity needs to be known either *a priori* or by measurements with external sensors.

For a more detailed description on how the kinematics of the RCM is computed, please refer to Appendix A.

2) *Path Tracking*: The secondary subtask is executing a desired motion, which, in cases like tumor resection, could be having the total system's end-effector follow a desired path. Given a reference Cartesian path $\tilde{\mathbf{h}}_p \in \mathbb{R}^{m_p}$, and its corresponding desired velocity $\dot{\tilde{\mathbf{h}}}_p$, with $m_p \leq 6$, from (7), the cost function associated to the secondary subtask can be expressed as $\|\dot{\tilde{\mathbf{h}}}_p - \mathbf{J}_{tot}\dot{\mathbf{q}}\|$. In this work, the path tracking is expressed in terms of the Cartesian position and orientation, therefore, $m_p = 6$.

B. Prioritized Motion Control

In order to guarantee accurate execution of the two prioritized subtasks, the HQP problem (10) is modified as follows:

$$\begin{aligned} &\text{for } n = 1, 2 \\ &\dot{\mathbf{q}}_n = \arg \min_{\dot{\mathbf{q}}} \frac{1}{2} \|\dot{\tilde{\mathbf{h}}}_n - \mathbf{J}_{h,n}\dot{\mathbf{q}}\|^2 + \frac{1}{2} \|\Lambda\dot{\mathbf{q}}\|^2 \\ &\text{s.t. } \mathbf{J}_{h,n-1}\dot{\mathbf{q}} = \mathbf{J}_{h,n-1}\dot{\mathbf{q}}_{n-1} \\ &\frac{\mathbf{q}_m - \mathbf{q}}{dt} \leq \dot{\mathbf{q}} \leq \frac{\mathbf{q}_M - \mathbf{q}}{dt} \end{aligned} \quad (12)$$

where $n = 1$ corresponds to guaranteeing RCM motion ($\mathbf{J}_{h,1} = \mathbf{J}_{RCM}$) and $n = 2$ is for the path tracking ($\mathbf{J}_{h,2} = \mathbf{J}_{tot}$). For the primary RCM subtask, $\mathbf{J}_{h,0} = \mathbf{0}$ and only joint limits are considered. $\Lambda = \text{diag}(\lambda_k, \lambda_u)$ is a diagonal weighting matrix, which is used to reduce joint velocities when the systems are close to singularity. The entries of $\lambda_k \in \mathbb{R}^7$ and $\lambda_u \in \mathbb{R}^5$ are chosen such that

$$\lambda_i = \begin{cases} (1 - (\frac{\sigma_m}{\sigma_{\text{thresh}}})^2)\lambda_M, & \text{if } \sigma_m \leq \sigma_{\text{thresh}} \\ 0, & \text{otherwise} \end{cases} \quad (13)$$

with σ_m being the smallest eigenvalue of the corresponding robot Jacobian. σ_{thresh} and λ_M are user specified values.

To reduce the tracking errors of both the primary and secondary subtasks, the desired tracking velocities are modified according to the closed-loop inverse kinematics (CLIK) [38] method as

$$\begin{cases} \dot{\tilde{\mathbf{h}}}_1 = \dot{\tilde{\mathbf{h}}}_{RCM} + \mathbf{K}_{RCM}e_{RCM} \\ \dot{\tilde{\mathbf{h}}}_2 = \dot{\tilde{\mathbf{h}}}_P + \mathbf{K}_P e_P \end{cases} \quad (14)$$

with $e_{RCM} \in \mathbb{R}^2$ and $e_P \in \mathbb{R}^6$ being the RCM and the tip position and orientation (expressed in angle-axis formulation) tracking errors. The tracking errors can be measured by external sensors, if present, or estimated from the kinematic model, otherwise. The matrices $\mathbf{K}_{RCM}$ and $\mathbf{K}_P$ are positive definite diagonal matrices of user's defined gains.

C. Configuration Optimization

Fig. 6 shows the flow chart of the procedure to find the optimal surgical robotic instrument's base configuration.

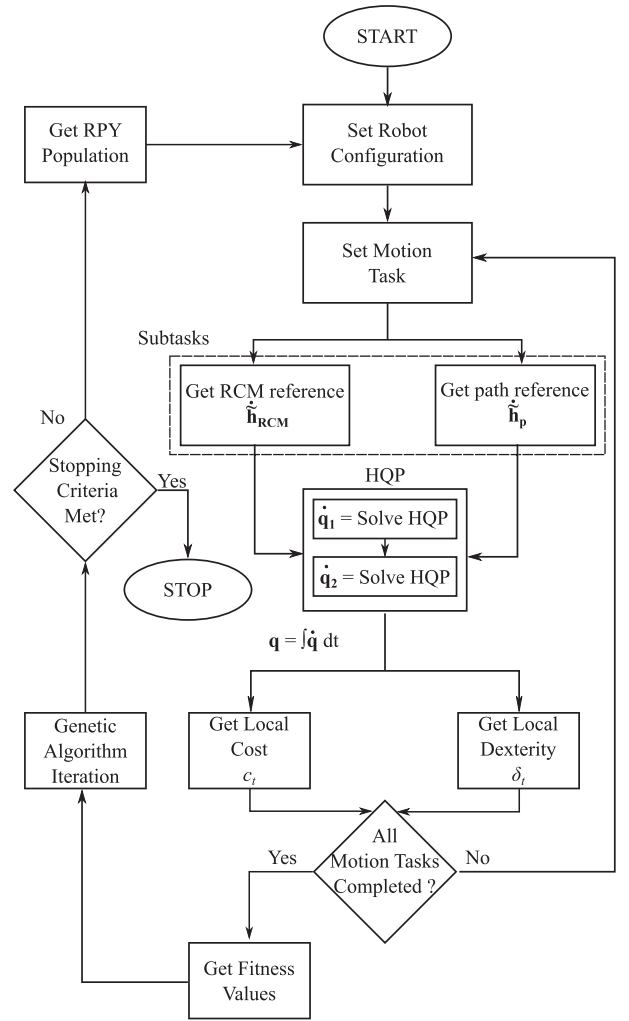


Fig. 6. Flow chart for the optimization of the robot configuration.

The GA algorithm presented in Section III-B is employed to find the optimal Micro-IGES pose with respect to the KUKA robot. Therefore, the optimization variable is the Micro-IGES base to KUKA robot transformation matrix kT_b , as described in Section II-C and shown in Fig. 7, which is defined as

$${}^kT_b = \begin{bmatrix} {}^kR_b & {}^kP_{b,k} \\ \mathbf{0} & 1 \end{bmatrix}. \quad (15)$$

Since the Micro-IGES shaft is rigid, it follows that ${}^kP_{b,k} = {}^kR_b \mathbf{r}$, with $\mathbf{r} = [0, 0, 0.27]^T$ in meters, due to the length of the shaft. Consequently, the optimization problem can be formulated in terms of kR_b only.

At each iteration, the GA generates a population of possible configurations for the surgical robotic instrument. At each configuration, a desired motion task is solved. In this work, we are specifying three motion tasks, defined by $\mu = 1, 2, 3$ and simulating the resection of three circular tumors at different locations, but using the same insertion point. Fig. 8 shows some snapshots of the different motion tasks, for two different base configurations. The purpose of using three different tumors is to make the optimization framework generalizable to different

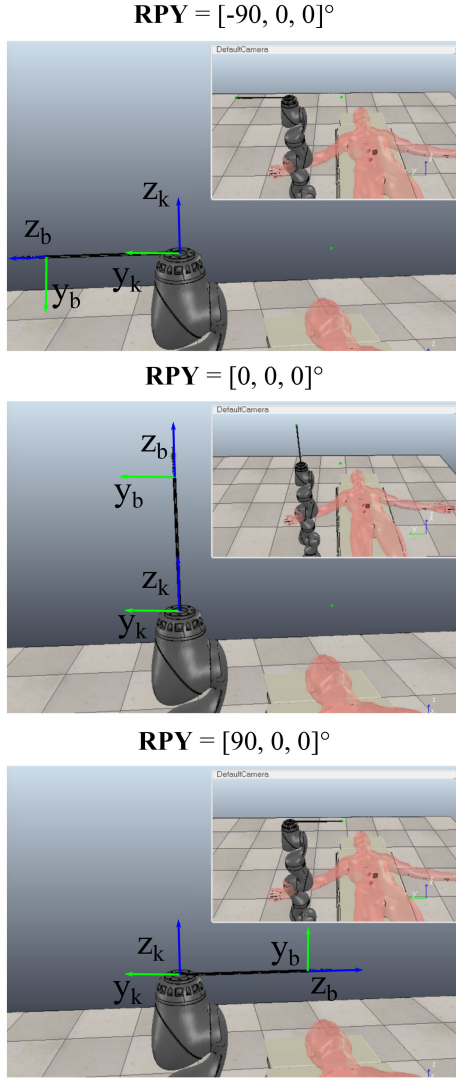


Fig. 7. macro-micro manipulator's structure at different surgical robotic instrument's base configurations, defining the optimization variable kT_b .

motion tasks. If only one motion task is assigned, the optimal configuration may not be suitable for a different task. As described in Section IV-A, each motion task is divided into two subtasks to guarantee the RCM motion and accurate path tracking, and solved at discrete time instants by means of (12). At each instant, the cost and dexterity measures are computed, as described in the following. These values will then be used in the fitness function for the GA. Once a motion task is completed, the system moves on to the next one, and when all the motion tasks are completed, the total fitness function is computed and passed to the GA for the generation of the next population. The procedure keeps running until the GA meets the stopping criteria.

1) *GA Specifications*: For the optimal configuration problem, the *Population* consists of individuals described by a set of RPY angles, which define kR_b . Because of the robot design, the Roll and Pitch are limited between $\pm 90^\circ$, whereas the yaw can vary between $\pm 180^\circ$. Moreover, each RPY angle could be considered to be continuous, meaning $RPY \in \mathbb{R}^3$.

TABLE I
GA PROPERTIES AND THEIR ASSIGNED VALUES

Property	Value
Creation function	<i>Uniform</i>
Fitness scaling function	<i>Rank</i>
Selection function	<i>Tournament</i>
Elite children factor	0.2
Crossover factor	0.8
Crossover function	<i>Heuristic</i>
Mutation function	<i>Gaussian</i>

However, fractions of angles may be difficult to achieve in a real world due to manufacturing tolerances. Therefore, in this work, we consider each angle to be a discrete integer number and $RPY \in \mathbb{Z}^3$, which guarantees a resolution of 1° . The optimization problem then becomes a mixed-integer GA.

Due to this discretization, the optimal configuration may be retrieved by simply searching throughout the domain of the optimization variables. This, however, would require $181^2 \times 361 = 11\,826\,721$ explorations, which might be unreasonable to solve. The GA, instead, may find the best solution by evaluating smaller sets, since at each generation, a new *Population* is created. If available, *a priori* knowledge about the robot model could be exploited to reduce the search space.

Table I reports the values of the properties used to set up the GA optimizer in MATLAB, according to the description presented in Section III-B. For the generation of the population, *Uniform* function is used, which creates a random initial population with a uniform distribution, with the population always within the imposed range.

MATLAB *Rank* and *Tournament Selection* functions are used to rank each individual based on the value of its fitness function and to select the parents from the population. The use of the selection function is imposed by MATLAB, given the mixed-integer nature of the optimization.

For generating the children from the individuals in the current populations, 20% of all the individuals are chosen to be passed directly to the next population as *Elite Children*. Among the remaining individuals, 80% are obtained from the *Crossover Function*, using the *Heuristic* function, and 20% from the *Mutation Function*, using the *Gaussian* function to generate random children.

Finally, for the stopping criteria, the maximum number of generations is set to 100, and the maximum number of *Stall Generations* is set to 10.

2) *Fitness Function*: The GA will find the optimal RPY that minimize a certain fitness function. This function should take into consideration different factors such as the RCM motion, the tip positioning, and the robot dexterity in performing the desired motion tasks.

To perform the path tracking for each motion task $\mu = 1, 2, 3$ by using the stack of tasks described in Section IV-A, the path is discretized in N_t time steps, and (12) is used to find the joint commands at each time step. It is worth mentioning that the number of time steps is kept the same for each mounting configuration. The tracking accuracy can, therefore, be expressed

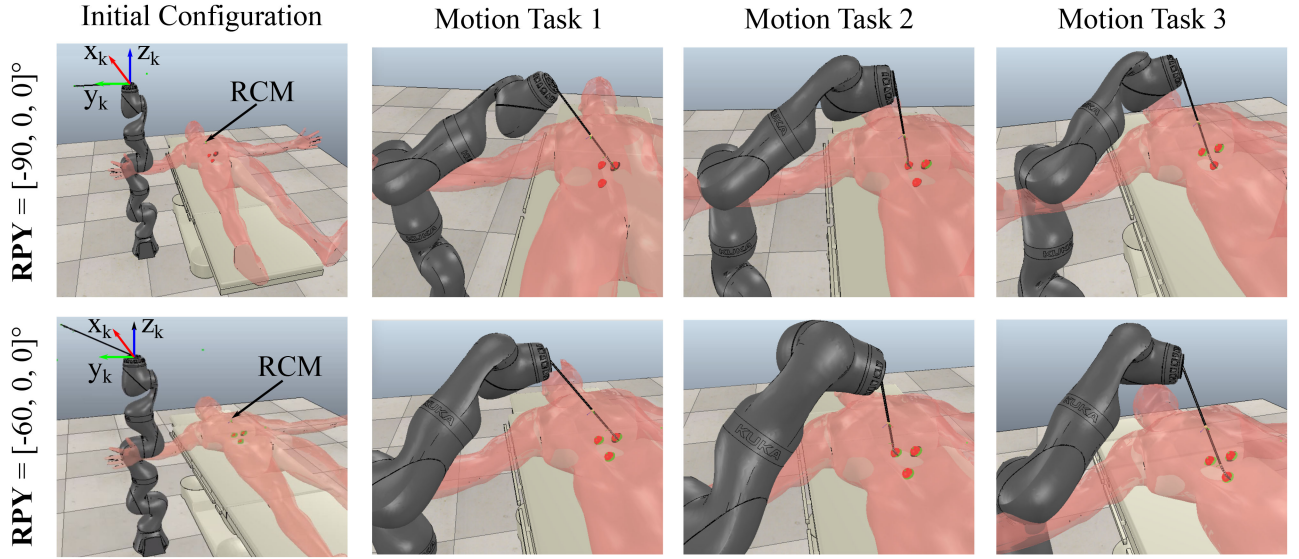


Fig. 8. Snapshots of the three motion tasks ($\mu = 1, 2, 3$). In these cases, two exemplary configurations are shown.

in terms of the value of the cost function achieved by solving (12). Assuming that $\dot{\mathbf{q}}^*$ is the optimal joint commands from (12) when both the RCM and the path tracking subtasks are solved, the cost at each time step $t = 0 \dots N_t$ is computed as

$$c_t = \frac{1}{2} \|\dot{\mathbf{h}}_{\text{RCM}} + \mathbf{K}_{\text{RCM}} \mathbf{e}_{\text{RCM}} - \mathbf{J}_{\text{RCM}} \dot{\mathbf{q}}^*\| + \frac{1}{2} \|\dot{\mathbf{h}}_P + \mathbf{K}_P \mathbf{e}_P - \mathbf{J}_{\text{tot}} \dot{\mathbf{q}}^*\|, \quad (16)$$

with each $c_t \in [0, \infty]$. This formulation allows to also take into account the positioning errors at each instant, and at best, each $c_t = 0$. For each motion task and for each **RPY** set, the average $\bar{c}^\mu$ and the maximum c_M^μ cost can be obtained as $\bar{c}^\mu = \frac{1}{N_t} \sum_{t=0}^{N_t} c_t$ and $c_M^\mu = \max(c_0, \dots, c_{N_t})$.

On each motion task μ , the best **RPY** configuration is the one that minimizes both $\bar{c}^\mu$ and c_M^μ . In fact, just considering the average cost may not be enough, as some configurations may lead to high error peaks. Therefore, the cost fitness function, for each configuration, is defined as

$$f_c^\mu = \frac{1}{\bar{c}^\mu c_M^\mu} \geq 0 \quad (17)$$

which tends to penalize configurations with high cost peaks.

Beside the motion accuracy, it is also important to have configurations that improve the dexterity of the system. Generally, the manipulability ellipsoid [39] is used in robotics to assess the dexterity of a system. An ellipsoid is defined as $\mathbf{v}^T (\mathbf{H})^{-1} \mathbf{v} \leq 1$, with the singular values of $\mathbf{H}$ being the square of the length of its semiaxes. The manipulability ellipsoid is defined as the set of joint velocities such that $\dot{\mathbf{q}}^T \dot{\mathbf{q}} \leq 1$, which from the solution of (8), leads to $\dot{\mathbf{h}}^T (\mathbf{J} \mathbf{J}^T)^{-1} \dot{\mathbf{h}} \leq 1$. Different measures exist to assess the dexterity of a robot, as reported in [40]. This derivation, however, applies only if one single subtask is specified and does not consider joint limits. Therefore, in this work, the following modification is carried out.

If joint limits were not imposed, the solution of (12) would be the same as the one obtained from the gradient projection method (as in Section III-A) and would be defined by [41]

$$\dot{\mathbf{q}} = \mathbf{J}_1^\dagger \dot{\mathbf{h}}_1 + (\mathbf{J}_2 \mathcal{P}_1)^\dagger (\dot{\mathbf{h}}_2 - \mathbf{J}_2 \mathbf{J}_1^\dagger \dot{\mathbf{h}}_1) \quad (18)$$

with $\mathcal{P}_1 = \mathbf{I} - \mathbf{J}_1^\dagger \mathbf{J}_1$, and $\dot{\mathbf{h}}_1, \mathbf{J}_1$ correspond to the RCM subtask, and $\dot{\mathbf{h}}_2, \mathbf{J}_2$ to the path tracking subtask. In order to take into account joint limits, a penalization function is used [40]. Each Jacobian column $j = 1 \dots 12$ is penalized for each subtask $n = 1, 2$ such that

$$\mathbf{J}_{n,j} = s_j \mathbf{J}_{n,j} \quad s_j = \frac{1 - e^{-4K \frac{(q_{M,j} - q_j)(q_j - q_{m,j})}{(q_{M,j} - q_{m,j})^2}}}{1 - e^{-K}} \quad (19)$$

where K is a positive constant. This allows the contribution of a specific joint to be zero when it reaches its joint limits. From the definition of the manipulability ellipsoid, following the Jacobian penalization, and from (18), the multisubtask manipulability ellipsoid can be defined as

$$\dot{\mathbf{q}}^T \dot{\mathbf{q}} = \begin{bmatrix} \dot{\mathbf{h}}_1^T & \dot{\mathbf{h}}_2^T \end{bmatrix} \mathbf{H} \begin{bmatrix} \dot{\mathbf{h}}_1 \\ \dot{\mathbf{h}}_2 \end{bmatrix} \quad (20)$$

with $\mathbf{H} \in \mathbb{R}^{8 \times 8}$, since it includes the 2-D RCM subtask and the 6-D path tracking subtask. If only one single subtask is specified, this formulation reduces to the traditional manipulability ellipsoid formulation. To assess the system dexterity, the inverse condition number is used [40]. Due to the time discretization of the motion, at each time step, the dexterity measure is defined as

$$\delta_t = \frac{\sigma_{H,m}}{\sigma_{H,M}} \quad (21)$$

with $\delta_t \in [0, \infty]$ and $\sigma_{H,m}, \sigma_{H,M}$ are the minimum and maximum singular values of $\mathbf{H}$. For each motion task and for

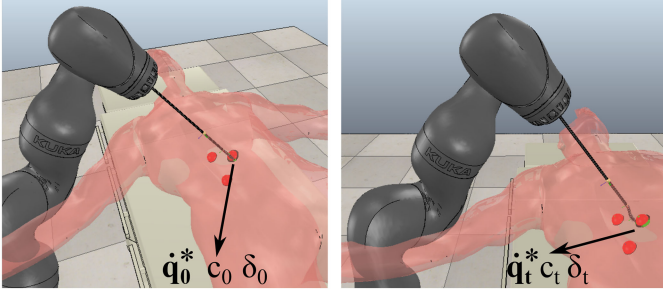


Fig. 9. Example of the cost and dexterity computation for each time step $t = 0, \dots, N_t$ of each motion task, for a given **RPY** robot configuration.

each **RPY**, $\bar{\delta}^\mu = \frac{1}{N_t} \sum_{t=0}^{N_t} \delta_t$ and $\delta_m^\mu = \min(\delta_0, \dots, \delta_{N_t})$ are computed. To choose the **RPY** with best dexterity, the dexterity fitness function is formulated as

$$f_d^\mu = \bar{\delta}^\mu \delta_m^\mu \geq 0. \quad (22)$$

This function is again computed on each motion task μ and for each configuration. Configurations with high dexterity, will lead to high $\bar{\delta}$, δ_m , and f_d . Fig. 9 shows two examples of the cost and dexterity computation for a given motion task and a given **RPY** configuration.

In order to find the **RPY** configuration with the best dexterity and smallest cost, thus maximizing both f_c^μ and f_d^μ on each motion task, the overall motion task fitness function is defined as $f^\mu = f_c^\mu f_d^\mu$. Finally, the total fitness function that the GA will minimize is set to be

$$F = -\frac{1}{3} \sum_{\mu=1}^3 f^\mu \quad (23)$$

with the negative sign being added to ensure the minimization of the fitness function.

3) *Resilience to Errors*: One issue that may arise from this optimization is that the optimal configuration identified by the solver may reside in a region where small deviations from the optimal may lead to much worse performance, yielding higher costs and lower dexterities. This is what could happen in case of positioning errors during the assembly on the real system. In order to overcome it, the following adjustment is made.

Given a robot configuration **RPY**⁰, a neighboring region to this configuration is considered. This region is defined by a minimum (l_m) and maximum (l_M) size along each dimension. Due to the discretization of the angles, the size of the region is computed as $l = l_m + n_l$, with $n_l \in \mathbb{N} \cap [0, N_l]$, $N_l = l_M - l_m \in \mathbb{N}^+$ being the maximum number of intervals. This means that to each initial configuration corresponds a region $\mathcal{R}$ of $N_l * 2^{\dim}$ neighboring configurations **RPY**^s, with $\dim = 1, 2, 3$, depending on how many angles are optimized. Fig. 10 shows an example of the neighboring region when $\dim = 1$ (only one angle optimized) and $\dim = 3$.

For each neighboring configuration **RPY**^s in $\mathcal{R}$, each motion task μ is solved and $\bar{c}$, c_M , $\bar{\delta}$, and δ_m are computed. Then, the deviation of the costs and dexterities of each neighboring configuration with respect to the initial configuration **RPY**⁰ is computed. The initial configuration is associated

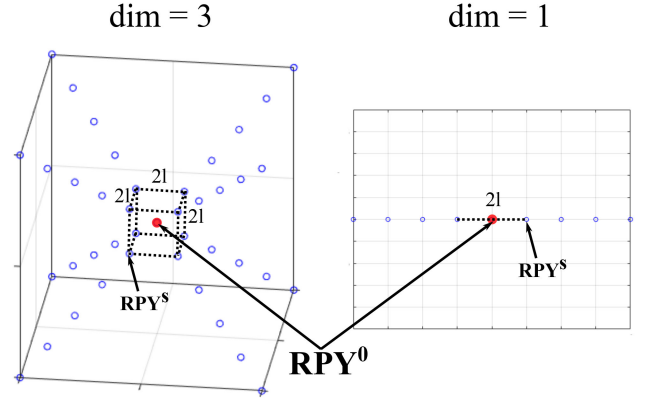


Fig. 10. Neighboring region $\mathcal{R}$ for a given configuration **RPY**⁰ when optimizing for one angle ($\dim = 1$) and for all RPY angles ($\dim = 3$).

Algorithm 1: Algorithm for Penalization of the Costs and Dexterities Based on Neighboring Region. This is Implemented for Each Motion Task μ .

```

1: function  $\bar{c}^\mu, c_{M,0}^\mu, \bar{\delta}^\mu, \delta_{m,0}^\mu \leftarrow \text{RegionPenalty}$ 
   (RPY0,  $N_l$ ,  $\dim$ )
  ▷ Get neighboring region
2:  $\mathcal{R} \leftarrow \text{getRegion}(\text{RPY}^0, N_l, \dim)$ 
3:  $S = N_l 2^{\dim}$  ▷ Region size
  ▷ Get values for current configuration
4:  $\bar{c}_0^\mu, c_{M,0}^\mu, \bar{\delta}_0^\mu, \delta_{m,0}^\mu \leftarrow \text{SolveMotionTask}(\text{RPY}^0)$ 
5:  $s = 1$ 
6: for RPYs  $\in \mathcal{R}$  do
7:    $\bar{c}_s^\mu, c_{M,s}^\mu, \bar{\delta}_s^\mu, \delta_{m,s}^\mu \leftarrow \text{SolveMotionTask}(\text{RPY}^s)$ 
8:    $s = s + 1$ 
9: end for
10:  $\Delta_1^\mu = \max(|\bar{c}_1^\mu - \bar{c}_0^\mu|, \dots, |\bar{c}_S^\mu - \bar{c}_0^\mu|)$ 
11:  $\Delta_2^\mu = \max(|c_{M,1}^\mu - c_{M,0}^\mu|, \dots, |c_{M,S}^\mu - c_{M,0}^\mu|)$ 
12:  $\Delta_3^\mu = \max(|\bar{\delta}_1^\mu - \bar{\delta}_0^\mu|, \dots, |\bar{\delta}_S^\mu - \bar{\delta}_0^\mu|)$ 
13:  $\Delta_4^\mu = \max(|\delta_{m,1}^\mu - \delta_{m,0}^\mu|, \dots, |\delta_{m,S}^\mu - \delta_{m,0}^\mu|)$ 
  ▷ Penalize current configuration values
14:  $\bar{c}_0^\mu = \bar{c}_0^\mu \Delta_1$ 
15:  $c_{M,0}^\mu = c_{M,0}^\mu \Delta_2$ 
16:  $\bar{\delta}_0^\mu = \frac{\bar{\delta}_0^\mu}{\Delta_3}$ 
17:  $\delta_{m,0}^\mu = \frac{\delta_{m,0}^\mu}{\Delta_4}$ 
   return  $\bar{c}_0^\mu, c_{M,0}^\mu, \bar{\delta}_0^\mu, \delta_{m,0}^\mu$ 
18: end function

```

with $\bar{c}_0^\mu, c_{M,0}^\mu, \bar{\delta}_0^\mu, \delta_{m,0}^\mu$. Since the optimal **RPY** configuration should reside in a region with small costs and dexterities variations, the maximum deviations $\Delta_1^\mu, \Delta_2^\mu, \Delta_3^\mu, \Delta_4^\mu$ in the region are computed and used to penalize the cost and dexterity values of the current initial configuration. The penalized costs and dexterities are then used to compute the fitness functions F for the GA. In this way, configurations leading to large deviations will be associated with higher penalized costs and smaller penalized dexterities. Algorithm 1 describes the penalization process.

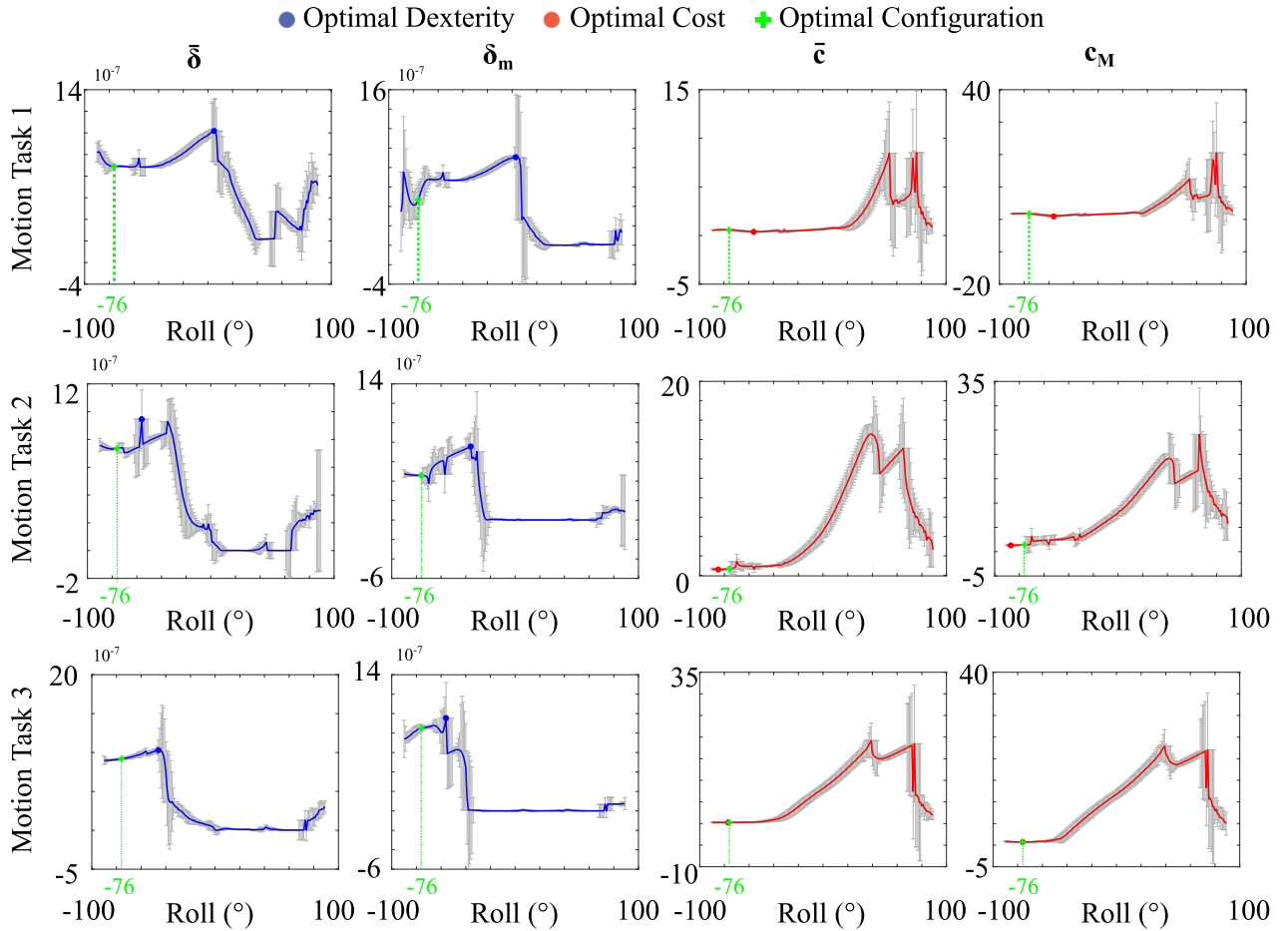


Fig. 11. Plots of the costs and dexterities for the optimization of only the Roll angle in the different motion tasks $\mu = 1, 2, 3$ in simulation. The gray bars are the maximum deviations for the costs and dexterities in the neighboring region $\mathcal{R}$ at each robot configuration.

V. RESULTS

In this section, the results of the proposed method are reported, both for a simulated test and real-world experiments.

A. Simulation Results

The proposed approach is employed to find the optimal configuration when only optimizing for one angle and when optimizing for the whole RPY set of angles. In the simulation test, the macro-micro manipulator needs to cut three circular tumors of 15 mm in radius in different locations with respect to the KUKA base frame. For the tracking subtask, the end-effector tip is required to be always perpendicular to the tumors' plane, and for all the tumors, the same insertion point is used, specified with respect to the KUKA base frame.

1) *Roll Optimization*: If only one single angle is required to be optimized, the GA optimization may be redundant. For this test, only the Roll angle is optimized, with the pitch and yaw set to 0° . The Roll angle about the x -axis of KUKA end-effector's frame, due to the robot design, needs to be limited in the range $[-90, 90]^\circ$. If a resolution of 1° is wanted, then the Roll angle can be discretized in 181 points. A simple space search would be able to find the optimal solution. Using the GA, with a population of 181 individuals, the algorithm still finds the same optimal

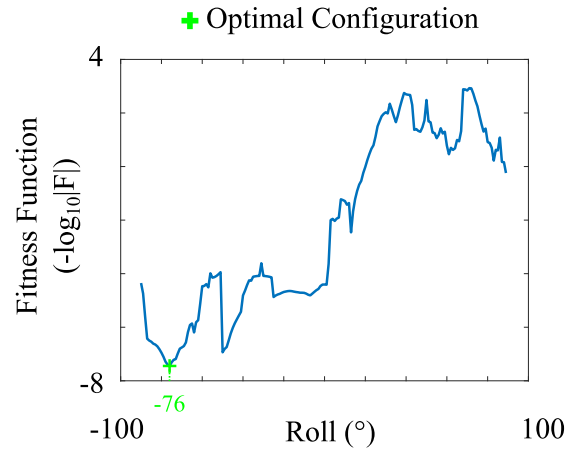


Fig. 12. Fitness function F in the logarithmic scale for the Roll optimization in the simulation test.

solution, yet in more iterations (11 in total), since it needs to meet the stopping criteria, such as reaching the maximum number of *Stall Generations*.

Fig. 11 shows the optimization results for costs ($\bar{c}$, c_M) and dexterities ($\bar{\delta}$, δ_m) on the three motion tasks and Fig. 12 shows the distribution of the fitness function, in logarithmic scale,

TABLE II
COST AND DEXTERITY VALUES FOR THE OPTIMAL ROLL CONFIGURATIONS
WHEN PERFORMING THE THREE MOTION TASKS $\mu = 1, 2, 3$

Optimal roll angle = -76°				
Motion task	$\bar{c}$	c_M	$\bar{\delta} (\cdot 10^{-6})$	$\delta_m (\cdot 10^{-6})$
$\mu = 1$	0.58	1.76	0.69	0.45
$\mu = 2$	0.66	1.42	0.74	0.46
$\mu = 3$	0.25	0.72	0.91	0.85
Fitness (F)				
-2.79×10^7				

for the different Roll angles. In this scenario, the size of the neighboring region is chosen to range from 1° to 5° . The optimal Roll angle, allowing to have the best performance over the three motion tasks, results to be $\text{Roll}_{\text{opt}} = -76^\circ$ to which corresponds a fitness function value of $-\log_{10}(|F|) = -7.45$. Table II reports the costs and dexterities when using the optimal configuration Roll_{opt} . The low values of the dexterity measures are due to the constraint imposed by the RCM: if the same motion task was specified without taking into account tracking the RCM, larger dexterity values would be achieved.

The optimal configuration Roll_{opt} found by the GA optimizer does not always lead to optimal costs or dexterities. This is mainly due to two reasons. First, the fitness function is a combination of all the measures used (average and minimum dexterity, average and maximum cost), and minimizing for one measure may lead to a deterioration of another measure. A second reason is the penalizations introduced as described in Section IV-C3. For instance, in motion task 1, the Roll angle of 5° leads to the highest average dexterity. However, this configuration is not resilient to errors: small deviations from it will lead to great dexterity losses. The same applies to the other measures. The optimal configuration, instead, always resides within a region with small deviations. If the resilience to the errors was not considered in the optimization framework, the optimal Roll angle would be -56° . Even though this configuration might have higher dexterities in each of the motion tasks, compared to Roll_{opt} , it does not reside within a safe region.

It is interesting to notice that the configurations with highest dexterities and smaller costs are those with negative Roll angles. Indeed, Fig. 13 shows that the configurations with positive Roll, have larger number of joints reaching their limits during the motion, thus limiting the system's capabilities in performing the desired motion tasks.

2) *RPY Optimization*: If the whole set of optimal RPY angles is sought, the GA becomes more useful. In fact, a space search would require to go through millions of configurations, thus requiring long computational time. For the optimal *RPY*, a population of 500 individuals was used and the size of the neighboring region was set to 5° . The GA converged to the optimal configuration $\text{RPY}_{\text{opt}} = [-85, -67, 87]^\circ$, corresponding to $-\log_{10}(|F|) = -8.20$, in 33 iterations.

Table III reports the cost and dexterities associated with the optimal RPY configuration on the three different motion tasks, whereas Fig. 14 shows the approximated distribution of the costs and dexterities for different RPY configurations in the different motion tasks, and Fig. 15 the absolute value (in logarithmic

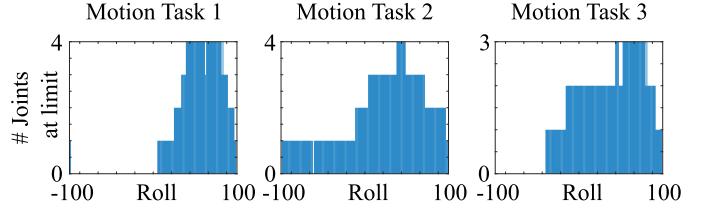


Fig. 13. Number of joints reaching their limits in the different motion tasks $\mu = 1, 2, 3$ at the different Roll configurations.

TABLE III
COST AND DEXTERITY VALUES FOR THE RPY OPTIMIZATION IN THE THREE
MOTION TASKS $\mu = 1, 2, 3$

Optimal configuration $\text{RPY} = [-85, -67, 87]^\circ$				
Motion task	$\bar{c}$	c_M	$\bar{\delta} (\cdot 10^{-6})$	$\delta_m (\cdot 10^{-6})$
$\mu = 1$	0.25	0.53	0.90	0.90
$\mu = 2$	0.32	0.95	0.80	0.50
$\mu = 3$	0.17	0.42	1.30	1.20
Fitness (F)				
$-1.59 \cdot 10^8$				

scale) of the fitness function F . The plots were created using around 10 000 configurations and interpolating the values of the measures within these samples. Even though this is an approximation to visualize the results, the solution of the GA is still a more optimal configuration among the interpolated ones. This is indeed given by the fact that over the whole process, the GA explored 13 300 configurations, given that it ran for 33 iterations generating 400 new individuals at each iteration.

Also in this scenario, the optimal configuration identified by the GA algorithm does not always minimize each singular measure. However, the penalizations allow retrieving configurations with small costs and acceptable dexterities, lying in regions with small variations, thus being resilient to errors. Table IV reports the maximum deviations in the neighboring region for RPY_{opt} .

B. Experimental Results

In the real case scenario, the macro-micro manipulator needs to cut a single circular tumor of 10 mm in radius located at [39, 26, 10]mm from the KUKA base with a constant orientation, perpendicular to the tumor's plane, and with the RCM at [39, 26, 17]mm with respect to the KUKA base frame. The simulated environment was used as a testbench for our proposed framework to identify the optimal configuration for this new desired motion task. The purpose of the experiment is to compare the results with the original, currently set, surgical robotic instrument's base configuration $\text{RPY} = [-90, 0, -180]^\circ$, and the optimized one.

Fig. 16 shows the setup for the real-world experiments and the two connections tested. A human plastic phantom was used to simulate a human body and place the desired insertion point. Two 6-DOF NDI AURORA electromagnetic (EM) trackers, providing both 3-D position and 3-D orientation data, have been used: one at the surgical robotic instrument's base, to track the shaft; one at the robot's end effector to track the robot tip. The accuracy of the sensors is of 0.01 mm. The EM tracker on the surgical robotic instrument's shaft is used to estimate the

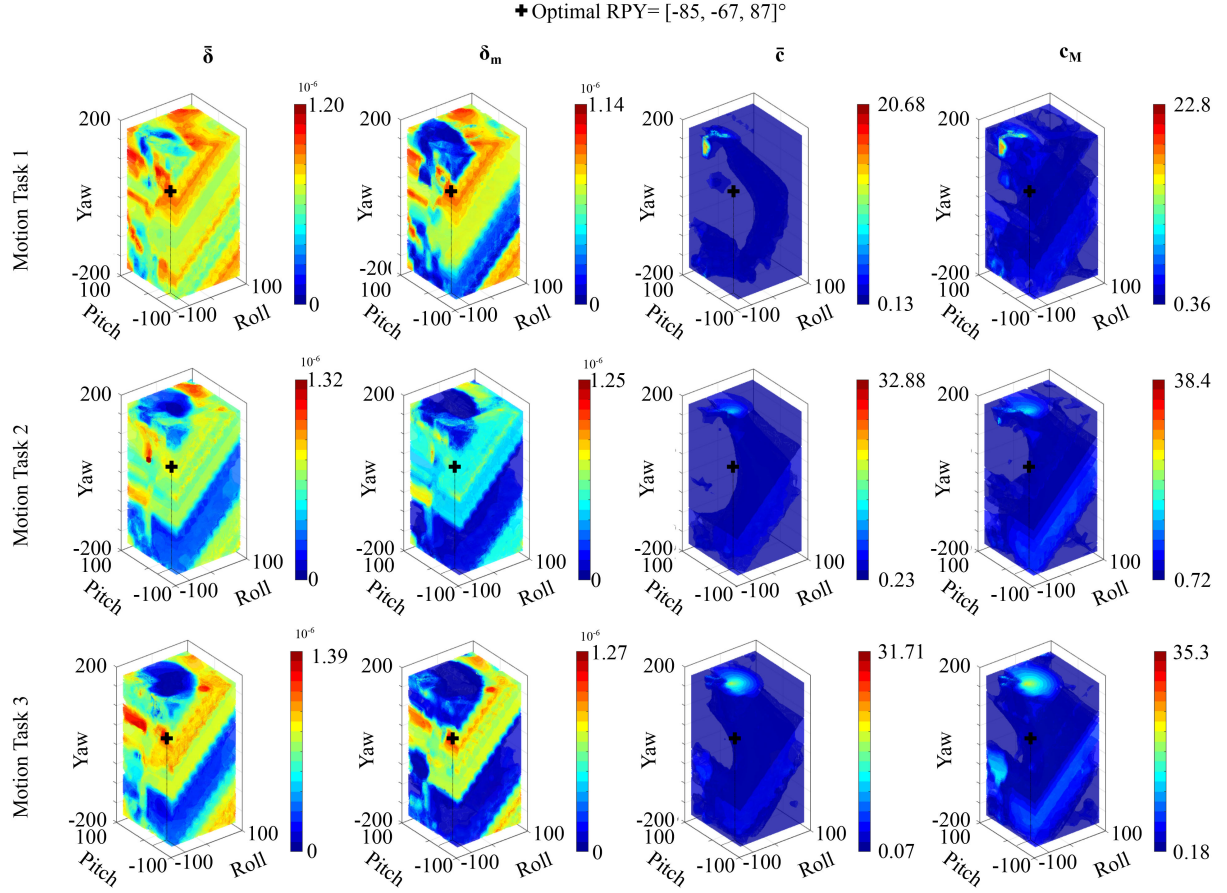


Fig. 14. Approximated costs and dexterities for the optimization of the RPY angles in the different motion tasks $\mu = 1, 2, 3$ for the simulation test.

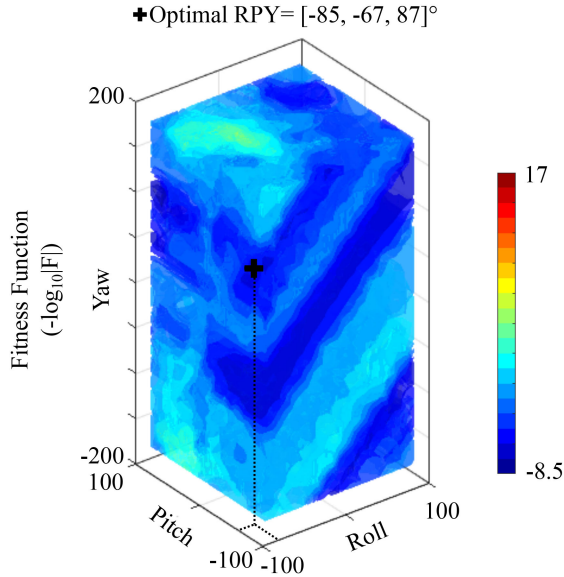


Fig. 15. Distribution of the absolute value of the fitness function F in the logarithmic scale over the RPY samples for the simulation test.

pose of the shaft, and from that, to measure the RCM tracking error along perpendicular directions of the shaft, as described in Appendix A.

TABLE IV
MAXIMUM DEVIATIONS (IN AVERAGE COST, MAXIMUM COST, AVERAGE DEXTERITY, MINIMUM DEXTERITY) IN THE NEIGHBORING REGION OF THE OPTIMAL RPY CONFIGURATION RPY_{opt} IN THE THREE DIFFERENT MOTION TASKS

Motion task	Maximum deviations for RPY_{opt}			
	Δ_1	Δ_2	$\Delta_3 (\cdot 10^{-6})$	$\Delta_4 (\cdot 10^{-6})$
$\mu = 1$	0.011	0.029	0.011	0.004
$\mu = 2$	0.028	0.061	0.071	0.017
$\mu = 2$	0.014	0.043	0.166	0.138

For the real test, we optimized only for the Roll and Pitch angles, since the Yaw angle can be achieved with the last joint of the KUKA robot, specifying a population of 500 individuals. Fig. 17 shows the convergence of the GA algorithm to the optimal solution $RPY_{\text{opt}} = [127, 41, 0]^\circ$.

The presence of model uncertainties in the Micro-IGES and some manual initial setting operations that need to be carried out (like bringing it back to its home configuration or the initial alignment with the hole) affect the performance of the system, even when repeating the same task. Therefore, the experiment was run five times in order to compare the macro–micro manipulator’s performance when using the original and the optimized configurations. Table V reports the performance measures for each test and the average ones over the five test runs: ($\bar{\epsilon}_P$)

TABLE V
EXPERIMENTAL RESULTS ON EACH TEST FOR BOTH THE ORIGINAL SURGICAL ROBOTIC INSTRUMENT'S BASE CONFIGURATION $RPY = [-90, 0, -180]^\circ$ AND THE OPTIMIZED ONE $RPY_{OPT} = [127, 41, 0]^\circ$

Test run	Configuration	$\bar{\delta} (\cdot 10^{-6})$	$\delta_m (\cdot 10^{-6})$	$\bar{\epsilon}_P$ (mm)	$\epsilon_{P,max}$ (mm)	$\bar{\epsilon}_\theta$ (rad)	$\epsilon_{\theta,max}$ (rad)	ϵ_{RCM} (mm)
1	Original	0.10	0.02	2.20	5.32	0.15	0.25	0.08
	Optimized	0.12	0.05	1.96	4.25	0.10	0.18	0.04
2	Original	0.06	0.02	1.80	6.41	0.12	0.21	0.05
	Optimized	0.13	0.004	2.21	4.56	0.17	0.38	0.04
3	Original	0.11	0.0003	2.13	6.28	0.13	0.21	0.07
	Optimized	0.09	0.002	2.15	5.31	0.12	0.24	0.03
4	Original	0.05	0.003	1.96	5.36	0.12	0.19	0.06
	Optimized	0.10	0.05	2.43	5.28	0.12	0.27	0.04
5	Original	0.05	0.02	2.03	5.17	0.12	0.18	0.07
	Optimized	0.11	0.008	2.36	5.31	0.18	0.39	0.05
Average	Original	0.07	0.01	2.02	5.71	0.13	0.21	0.06
	Optimized	0.11	0.02	2.22	4.94	0.14	0.29	0.04

and $\epsilon_{P,max}$ are the root mean squared error (RMSE) and the maximum value of the norm of the tip positioning error; ($\bar{\epsilon}_\theta$) is the RMSE of the orientation error (expressed in angle axis notation), and ($\epsilon_{\theta,max}$) the maximum norm of the angular error; (ϵ_{RCM}) the RMSE in tracking the RCM; and $\bar{\delta}$ and δ_m the average and minimum dexterity respectively.

Because of the calibration errors, inaccuracies in the kinematic models, and other nonlinearities not fully compensated (such as hysteresis and backlash in the Micro-IGES surgical robotic instrument), each test leads to slightly different results and to worse performance than in the simulated scenario. However, overall both configurations yield good tracking accuracy, with errors in the order of few millimeters. In terms of path tracking, the original configuration $RPY = [-90, 0, -180]^\circ$ achieves smaller average tip positioning and orientation errors, yet larger errors occur in tracking the RCM, compared to the optimal configuration $RPY = [127, 41, 0]^\circ$. With regards to the dexterity, the optimal configuration ensures higher dexterity measures, resulting also from a lower number of joints reaching their limits. Due to the chosen fitness function, the optimal configuration ensures a good tradeoff between tracking accuracy and dexterity. Our testbench simulation environment would predict a fitness function $F = -11.7 \times 10^6$ for the optimized configuration, compared to $F = -18.8 \cdot 10^2$ for the original one. Yet, the inherent modeling errors do not allow achieving those performance.

VI. CONCLUSION

Achieving safe and accurate execution of a specified surgical task, while satisfying the RCM and additional constraints, is of uttermost importance in MIRS. The control technique should be capable of ensuring high accuracy, while not violating the imposed constraints. The proposed HQP framework can be implemented as it allows solving prioritized subtasks while additional constraints are imposed.

In order to guarantee RCM motion, macro serial-link manipulators can be employed in conjunction with microsurgical robotic instruments. However, the connection and placement of the surgical instrument on the macromanipulator may affect the performance of the system, while performing the specified tasks. Therefore, an optimal configuration for the connection should be sought.

In this article, we presented a novel framework to identify the optimal configuration of the base of a surgical robotic instrument with respect to the serial-link manipulator's end-effector frame preoperatively. This framework exploited the capabilities of the HQP control strategy that allowed a highly redundant robotic system, consisting of a macro serial-link KUKA manipulator and the Micro-IGES surgical robotic instrument, to effectively perform specified motion tasks, while guaranteeing the RCM and additional constraints on the robot joint limits. To identify the optimal base configuration, the GA was employed, thanks to its capabilities to explore large search spaces.

Simulation results showed that some mounting configurations do not allowed achieving a high performance in terms of dexterity and accuracy, thus justifying the need to find an optimal configuration for the connection between the two robotic systems. The proposed framework proved effective in finding optimal configurations, both when optimizing only for the Roll angle and for the whole RPY angles, which guarantee small tracking errors of the desired paths, satisfaction of the RCM motion constraint, and maximization of the dexterity of the whole macro-micro manipulator. Moreover, the proposed strategy to make the optimal solution resilient to possible assembly errors was shown to be essential to ensure that the optimal configuration resides within a region where small variations in the tracking costs and dexterities occur.

Experimental results were carried out to test the performance on the real robotic system. Even though the simulation environment was predicting good performance for the optimized configuration, the inherent calibration and modeling errors in the system still affected the resulting performance.

In this work, the proposed framework was employed to only optimize for the surgical robotic instrument's base configuration, while considering the base of the serial-link manipulator, the insertion point, and the RCM fixed. However, the framework can be generalized to find the optimal insertion point placement or to include the serial-link manipulator's base configuration in the optimization. Moreover, for the RPY optimization, a population of 500 individuals was used. A larger number of individuals may lead to a better optimal solution, even though the computational time will be highly increased. The main limitation of the framework was that it is task-specific, therefore, a new optimal configuration may need to be recomputed and a new

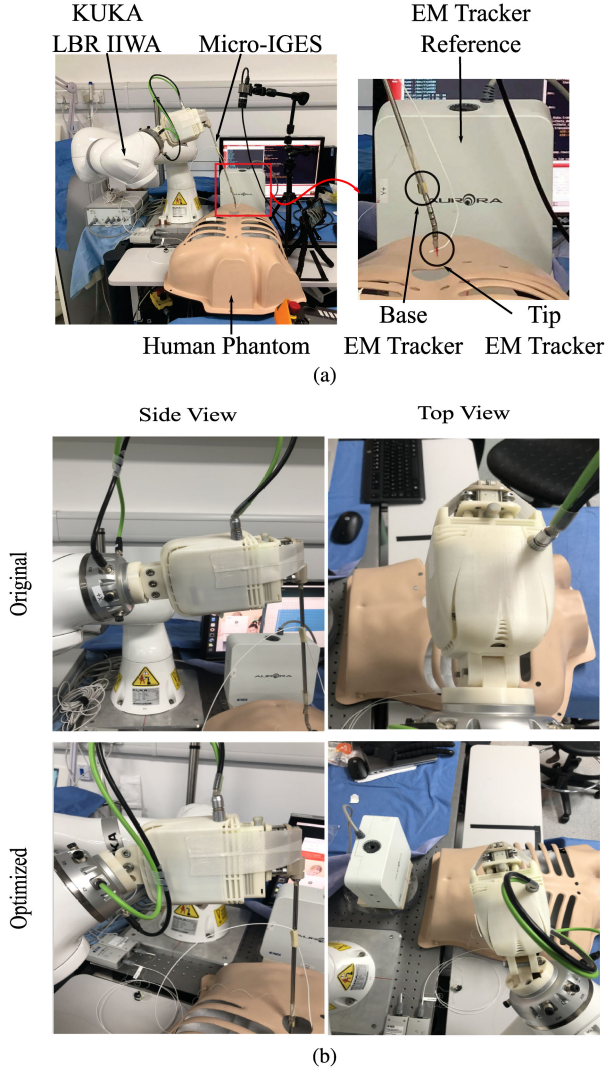


Fig. 16. Overview of the setup for the real experiments. (a) Setup. (b) Original $RPY = [-90, 0, -180]^\circ$ and optimized $RPY_{opt} = [127, 41, 0]^\circ$ connections between the two robots. (a) The setup for the real experiments. (b) The original and optimized RPY connections.

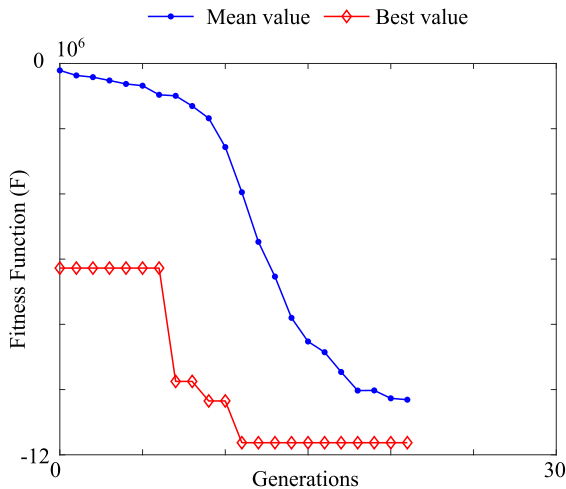


Fig. 17. Convergence of the GA for the optimization in the real scenario. The blue dots are the mean value of the fitness function F within the population at each generation; and the red diamonds are the best fitness function value at each generation.

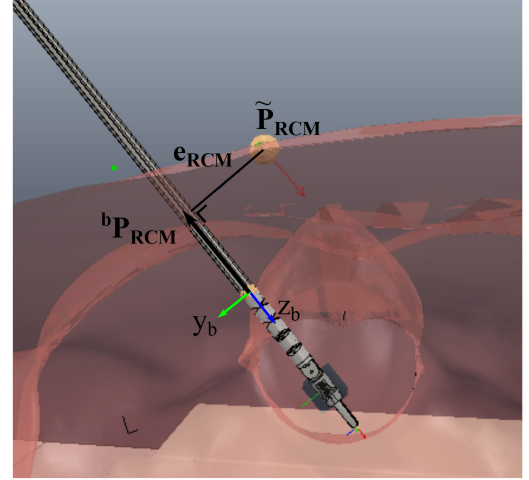


Fig. 18. Computation of the expected position of the RCM by projecting onto the surgical robotic instrument's shaft.

connection between the two robots designed when the surgical task changes. However, on one hand, since the optimization was run offline and thanks to the modern 3-D printing techniques, a new design can be easily made and used to improve the system's performance. On the other hand, the framework can be used to assess the capabilities of the macro-micro manipulator based on the available connections between the two robots for a given surgical task. To make the framework more generalizable, different surgical tasks can be specified so that the optimal configuration is the one that performs the best overall.

The simulation analysis and the experimental tests will be considered to design a new reconfigurable attachment for the correct assembly of the Micro-IGES surgical robotic instrument on the KUKA robot that could be applied to different tasks.

APPENDIX A RCM KINEMATICS

In order to compute the kinematics of the RCM, given the kinematic model of the whole macro-micro manipulator, it is sufficient to know the desired pose of the RCM, ${}^0\tilde{T}_{RCM}$, with respect to the macro-micro manipulator base frame. The estimated position of the RCM, however, needs to be projected onto the shaft of the surgical robotic instrument. Given the pose of the base of the surgical robotic instrument kT_b with respect to the KUKA end effector, the pose of the shaft can be computed as ${}^0T_b = {}^0T_k {}^kT_b$ and the pose of the RCM in the shaft frame will be ${}^bT_{RCM} = {}^0T_b^{-1} {}^0\tilde{T}_{RCM}$. In our case, z_b , the z -axis of the surgical robotic instrument's base, is aligned with the shaft, and since the RCM needs to be on the shaft, it is sufficient to project the actual RCM position on to the shaft to extract the expected RCM position as: ${}^bP_{RCM} = z_b^T {}^b\tilde{P}_{RCM} z_b$. The expected position of the RCM in the macro-micro manipulator's base frame will then be ${}^0P_{RCM} = {}^0P_k + {}^0R_k {}^kR_b {}^bP_{RCM}$. Fig. 18 shows a representation for the computation of the expected position of the RCM.

Thanks to this formulation, it is also possible to easily compute the error in tracking the RCM as $e_{RCM} = \tilde{P}_{RCM} - P_{RCM}$,

which is then used in the control scheme to improve the tracking accuracy. In the real experiments, the shaft pose is measured by the EM tracker at the surgical robotic instrument's base.

ACKNOWLEDGMENT

Data supporting this study are openly available on the Robot Intelligence Lab website <https://www.imperial.ac.uk/robot-intelligence/>. For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

REFERENCES

- [1] N. Aghakhani, M. Geravand, N. Shahriari, M. Vendittelli, and G. Oriolo, "Task control with remote center of motion constraint for minimally invasive robotic surgery," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 5807–5812.
- [2] R. C. O. Locke and R. V. Patel, "Optimal remote center-of-motion location for robotics-assisted minimally-invasive surgery," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 1900–1905.
- [3] W. Bai, Q. Cao, P. Wang, P. Chen, C. Leng, and T. Pan, "Modular design of a teleoperated robotic control system for laparoscopic minimally invasive surgery based on ROS & RT-Middleware," *Ind. Robot.*, vol. 44, no. 5, pp. 596–608, 2017.
- [4] B. Chen, Q. Cao, and W. Bai, "A design of surgical robotic system based on 6-DOF parallel mechanism," in *Proc. Int. Conf. Biol. Inf. Biomed. Eng.*, 2018, pp. 1–5.
- [5] C.-H. Kuo and J. S. Dai, "Robotics for minimally invasive surgery: A historical review from the perspective of kinematics," in *Proc. Int. Symp. Hist. Mach. Mechanisms*, Dordrecht, Netherlands: Springer, 2009, pp. 337–354.
- [6] M. M. Dalvand and B. Shirinzadeh, "Remote centre-of-motion control algorithms of 6-RRR parallel robot assisted surgery system (PRAMiSS)," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 3401–3406.
- [7] M. M. Marinho, M. C. Bernardes, and A. P. L. Bo, "Using general-purpose serial-link manipulators for laparoscopic surgery with moving remote center of motion," *J. Med. Robot. Res.*, vol. 1, no. 4, Dec. 2016, Art. no. 1650007.
- [8] W. Bai, Q. Cao, C. Leng, Y. Cao, M. G. Fujie, and T. Pan, "A novel optimal coordinated control strategy for the updated robot system for single port surgery," *Int. J. Med. Robot. Comput.-Assisted Surg.*, vol. 13, no. 3, Sep. 2017. [Online]. Available: <https://pubmed.ncbi.nlm.nih.gov/28758326/>
- [9] H. Sadeghian, F. Zokaei, and S. Hadian Jazi, "Constrained kinematic control in minimally invasive robotic surgery subject to remote center of motion constraint," *J. Intell. Robot. Syst.*, vol. 95, nos. 3–4, pp. 901–913, Sep. 2019.
- [10] C. D. Pham, F. Coutinho, A. C. Leite, F. Lizarralde, P. J. From, and R. Johansson, "Analysis of a moving remote center of motion for robotics-assisted minimally invasive surgery," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2015, pp. 1440–1446.
- [11] H. Azimian, R. V. Patel, and M. D. Naish, "On constrained manipulation in robotics-assisted minimally invasive surgery," in *Proc. IEEE RAS EMBS Int. Conf. Biomed. Robot. Biomechanics*, 2010, pp. 650–655.
- [12] J. Funda, R. Taylor, B. Eldridge, S. Gomory, and K. Gruben, "Constrained Cartesian motion control for teleoperated surgical robots," *IEEE Trans. Robot. Autom.*, vol. 12, no. 3, pp. 453–465, Jun. 1996.
- [13] A. Rocchi, E. M. Hoffman, D. G. Caldwell, and N. G. Tsagarakis, "OpenSoT: A whole-body control library for the compliant humanoid robot COMAN," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 6248–6253.
- [14] N. C. N. Doan and W. Lin, "Optimal robot placement with consideration of redundancy problem for wrist-partitioned 6R articulated robots," *Robot. Comput. Integr. Manuf.*, vol. 48, pp. 233–242, Dec. 2017.
- [15] D. Spensieri, J. S. Carlson, R. Bohlin, J. Kressin, and J. Shi, "Optimal robot placement for tasks execution," *Procedia CIRP*, vol. 44, pp. 395–400, Jan. 2016.
- [16] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the Nelder-Mead simplex method in low dimensions," *SIAM J. Optim.*, vol. 9, no. 1, pp. 112–147, 1998.
- [17] M. Schluter, S. Gerlach, C. Furweger, and A. Schlaefer, "Analysis and optimization of the robot setup for robotic-ultrasound-guided radiation therapy," *Int. J. Comput.-Assisted Radiol. Surg.*, vol. 14, no. 8, pp. 1379–1387, Aug. 2019.
- [18] R. Konietzschke, T. Ortmaier, H. Weiss, R. Engelke, and G. Hirzinger, "Optimal design of a medical robot for minimally invasive surgery," in *Jahrestagung der Deutschen Gesellschaft für Computer- und Roboterassistierte Chirurgie (CURAC)*, Nürnberg, Jan. 2003, pp. 1–8.
- [19] A. L. Trejos, R. V. Patel, I. Ross, and B. Kiaii, "Optimizing port placement for robot-assisted minimally invasive cardiac surgery," *Int. J. Med. Robot. Comput.-Assisted Surg.*, vol. 3, no. 4, pp. 355–364, Dec. 2007. [Online]. Available: <https://doi.wiley.com/10.1002/rcs.158>
- [20] X. Zhi, W. Bai, and E. M. Yeatman, "Kinematic parameter optimization of a miniaturized surgical instrument based on dexterous workspace determination," in *Proc. IEEE 6th Int. Conf. Adv. Robot. Mechatronics*, 2021, pp. 112–118.
- [21] J. Li, S. Wang, X. Wang, and L. Zhang, "Setup optimization for MIS robots with two-passive joints," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 2418–2423.
- [22] J. Shang *et al.*, "A single-port robotic system for transanal microsurgery-design and validation," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1510–1517, Jul. 2017.
- [23] A. Escande, N. Mansard, and P.-b. Wieber, "Hierarchical quadratic programming : Fast online humanoid-robot motion generation," *Int. J. Robot. Res.*, vol. 33, no. 7, pp. 1006–1028, 2014.
- [24] F. Cursi and G.-Z. Yang, "A novel approach for outlier detection and robust sensory data model learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Nov. 2019, pp. 4250–4257.
- [25] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC Press, 1994. [Online]. Available: <https://www.cds.caltech.edu/product/isbn/9780849379819/>
- [26] K. Leibbrandt *et al.*, "Effective manipulation in confined spaces of highly articulated robotic instruments for single access surgery," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1704–1711, Jul. 2017. [Online]. Available: <https://ieeexplore.ieee.org/document/7855734/>
- [27] F. Cursi, V. Modugno, and P. Kormushev, "Model predictive control for a tendon-driven surgical robot with safety constraints in kinematics and dynamics," in *Proc. Int. Conf. Intell. Robot. Syst.*, Jul. 2020, pp. 7653–7660. [Online]. Available: <https://spiral.imperial.ac.uk/handle/10044/1/80783>
- [28] P. Chang, "A closed-form solution for inverse kinematics of robot manipulators with redundancy," *IEEE J. Robot. Autom.*, vol. 3, no. 5, pp. 393–403, Oct. 1987.
- [29] J. Baillieul, "Avoiding obstacles and resolving kinematic redundancy," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1986, vol. 3, pp. 1698–1704. [Online]. Available: <https://ieeexplore.ieee.org/document/1087464/>
- [30] L. Sciavicco and B. Siciliano, "Solving the inverse kinematic problem for robotic manipulators," in *RoManSy 6*. Boston, MA, USA: Springer, 1987, pp. 107–114. [Online]. Available: https://link.springer.com/10.1007/978-1-4684-6915-8_9
- [31] A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 7, no. 12, pp. 868–871, Dec. 1977. [Online]. Available: <https://ieeexplore.ieee.org/document/4309644/>
- [32] F. Cursi, G. P. Mylonas, and P. Kormushev, "Adaptive kinematic modelling for multiobjective control of a redundant surgical robotic tool," *Robotics*, vol. 9, no. 3, p. 68, Aug. 2020. [Online]. Available: <https://www.mdpi.com/2218-6581/9/3/68>
- [33] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Mach. Learn.*, vol. 3, no. 2, pp. 95–99, 1988.
- [34] G. Renner and A. Ekárt, "Genetic algorithms in computer aided design," *Comput.-Aided Des.*, vol. 35, no. 8, pp. 709–726, Jul. 2003.
- [35] A. Schmitz, P. Berthet-Rayne, and G. Z. Yang, "Endoscopic Bi-manual robotic instrument design using a genetic algorithm," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, Nov. 2019, pp. 2975–2982.
- [36] P. K. Jamwal, S. Xie, and K. C. Aw, "Kinematic design optimization of a parallel ankle rehabilitation robot using modified genetic algorithm," *Rob. Auton. Syst.*, vol. 57, no. 10, pp. 1018–1027, Oct. 2009.
- [37] J. Han, W. K. Chung, Y. Youm, and S. H. Kim, "Task based design of modular robot manipulator using efficient genetic algorithm," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1997, vol. 1, pp. 507–512.
- [38] B. Siciliano, "A closed-loop inverse kinematic scheme for on-line joint-based robot control," *Robotica*, vol. 8, no. 3, pp. 231–243, Jul. 1990.
- [39] T. Yoshikawa, "Force control of robot manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2000, pp. 220–226.

- [40] D. Zhang, F. Cursi, and G.-Z. Yang, "WSRender: A workspace analysis and visualization toolbox for robotic manipulator design and verification," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3836–3843, Oct. 2019.
- [41] S. Chiaverini, G. Oriolo, and I. D. Walker, "Kinematically redundant manipulators," in *Springer Handbook of Robotics*. Berlin, Germany: Springer, 2008, pp. 245–268.



Francesco Cursi (Student Member, IEEE) received the B.S. in mechanical engineering from the Sapienza University of Rome, Rome, Italy, in 2014, and the M.S. degree in mechanical engineering from both Sapienza University of Rome and NYU Tandon School of Engineering, New York City, NY, USA, in 2016. He is currently working toward the Ph.D. degree with the Hamlyn Centre and Robot Intelligence Lab, Imperial College London, London, U.K.

His research interests include the intersection of robot control, machine learning, and optimization.



Weibang Bai (Member, IEEE) received the B.S. degree in mechanical engineering from Sichuan University, Chengdu, China in 2012, and the Ph.D. degree in robotics and mechanical engineering from Shanghai Jiao Tong University, Shanghai, China in 2019.

Since May 2019, he has been a Research Associate with the Hamlyn Centre and the Department of Computing, Imperial College London, London, U.K. His research interests include robotics, medical robotics, robotic design and intelligent control, robotic dexterous manipulation, and artificial intelligence.



Eric M. Yeatman (Fellow, IEEE) received the B.S. degree in physics from Dalhousie University, Halifax, NS, Canada, in 1983, the B.Eng. degree in engineering physics in 1985 from Technical University of Nova Scotia, Halifax, in 1985, the M.S. degree in physics in 1986 from Dalhousie University, in 1986, and the Ph.D. degree from Imperial College London, London, U.K., in 1989.

He has been a member of academic staff with Imperial College London since 1989, and a Professor of microengineering since 2005. He was appointed Head of the Department of Electrical and Electronic Engineering in September 2015. He has authored and co-authored more than 250 papers and patents on optical devices and materials, microelectromechanical systems, and other topics.

Dr. Yeatman was the recipient of the Royal Academy of Engineering Silver Medal in 2011, and become a Fellow of the Academy in 2012. He is also holds Fellowships of the Institution of Engineering and Technology, Institute of Materials, Minerals and Mining, and the City and Guilds Institute. He is the Chairperson of the Executive Board of Imperial-X. He has been a Principal or Co-investigator on more than 30 research projects and he has acted as a Design Consultant for several international companies, and technical advisory board member to two venture capital funds. His current research interests include energy sources for wireless devices, radio frequency and photonic microelectromechanical systems, sensor networks, and 2-D materials.



Petar Kormushev (Member, IEEE) received the M.S. degree in artificial intelligence and in bio- and medical informatics from Sofia University, Sofia, Bulgaria, in 2005 and 2006, respectively, and the Ph.D. degree from the Tokyo Institute of Technology, Tokyo, Japan, in 2009.

He led the Learning and Interaction Lab, Advanced Robotics Department, Italian Institute of Technology, Genoa, Italy and was a Visiting Senior Research Fellow with King's College London, London, U.K.

He is currently Senior Lecturer (equiv. to Associate Professor) in robotics with the Dyson School of Design Engineering and the Founder and Director of the Robot Intelligence Lab, Imperial College London, London. His research interests include machine learning and robot learning algorithms, especially reinforcement learning for intelligent robot behavior.