Date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

GlobDesOpt: A Global Optimization Framework for Optimal Robot Manipulator Design

FRANCESCO CURSI[1,2], WEIBANG BAI[1], ERIC M. YEATMAN[1], AND PETAR KORMUSHEV[2]
[1]Hamlyn Centre, Imperial College London, London, UK
[2]Robot Intelligence Lab, Imperial College London, London, UK

Corresponding author: F. Cursi (e-mail: f.cursi@ imperial.ac.uk).

This work was partially supported by EPSRC project EP/P012779/1.

ABSTRACT Robot design is a major component in robotics, as it allows building robots capable of performing properly in given tasks. However, designing a robot with multiple types of parameters and constraints and defining an optimization function analytically for the robot design problem may be intractable or even impossible. Therefore black-box optimization approaches are generally preferred.
In this work we propose GlobDesOpt, a simple-to-use open-source optimization framework for robot design based on global optimization methods. The framework allows selecting various design parameters and optimizing for both single and dual-arm robots. The functionalities of the framework are shown here to optimally design a dual-arm surgical robot, comparing the different two optimization strategies.

INDEX TERMS Robot design, Global optimization, Robot kinematics

I. INTRODUCTION

PROPERLY designing robots is a time-consuming and challenging task, which requires great expertise and knowledge on the field of application of the system. A good design should ensure good performance regardless of the task that it is given, otherwise different designs should be needed for different tasks. Thus, robotic design optimization with consideration of different types of parameters and constraints is of great significance and practicability.

Several measures can be employed to define an optimal design, such as the performance in executing a desired control task, accuracy in path tracking, and robot dexterity. However, analytically formulating an optimization function may be challenging or even impossible, especially for highly complex robotic structures. For these reasons, researchers have focused on black-box optimization methods, which do not require a close form of the optimization function, nor knowledge about its derivatives. For instance, Hassan et al. [1] employed Genetic Algorithm (GA) to optimize the design of a gripper; in [2] the optimal design of a 7DOF robot is studied using GA and considering different kinematic and dynamic measures; Particle Swarm Optimization (PSO) was used in [3] to optimally design a cable-driven robot and [4] employed Adaptive Simulated Annealing for the design of a concentric tube robot. Most of the works focus on the design of robots with just one arm. However, in fields like minimally invasive surgery, surgical robots are usually employed in a dual-arm configuration, both at the master and slave side. Optimally designing dual-arm robots is more challenging, as the interaction between the two arms should be considered. Because of the dual collaboration, performance indexes generally used for single-arm robots need to be adapted. Lee et al. [5] propose a dual arm manipulability measure specifically designed for multi-arm systems, and Torabi et

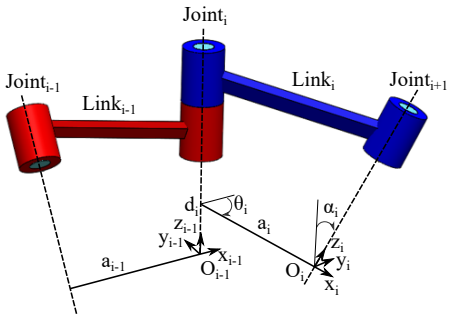


FIGURE 1: Representation of the Denavit-Hartenberg parameters

al. [6] extended it to master-slave teloperation. Only a few works in the literature have focused on the optimal design of dual-arm robots, such as [7] where PSO is used to design a dual-arm concentric-tube robot, or in [8] GA for a bi-manual articulated robot. However, both works focus on task specific applications.

Recently, in the field of machine learning, Bayesian Optimization (BO) has been gaining a lot of popularity and proved very efficient in solving black-box optimization problems. It is considered the state-of-the-art machine learning framework for optimization tasks in terms of data efficiency, and has been successfully applied in engineering, machine learning, and design [9], [10]. BO has also been widely used in robot control to optimize control parameters and policies [11]–[13] as it makes efficient use of data and experiments by learning a probabilistic surrogate model of the function to optimize, which is used for finding optimal parameters. By exploiting the learned model, BO requires fewer interactions (i.e., evaluations of the true objective function) than other optimization methods [11].

Lately, in [14] a method for optimizing the design of a single surgical robotic arm was proposed, yet the approach results to be little flexible, as it is very application specific, and time consuming. To the best of the authors' knowledge, no software is available online currently to simplify the robotic optimizing design process. We have therefore developed GlobDesOpt (available at https://github.com/cursi36/GlobDesOpt) a Matlab package for optimal robot design, based on three different solver (PSO, GA, and BO).

The contribution of this manuscript is two-fold:

- providing GlobDesOpt, a simple-to-use open-source Matlab package for optimal robot design;
- proposing two optimization strategies which can be applied to both single and dual-arm robots, with the dual-arm optimization including the dual-arm dexterity measure and a safety measure to maximize the dual-arm robot's dexterity while reducing the risk of collisions between the two arms.

GlobDesOpt allows selection of different design parameters, easy implementation of cost functions and models for different robotic structures. Currently, two main options are provided to optimize the design of robots in single or dual-arm configuration. In this work we compare the optimization results when employing the three different solvers, along with the optimization results when optimizing the design of a surgical slave robot for the two arms independently or simultaneously.

The manuscript is thus structured as follows. Section II introduces how robot design affects the robot's model and performance, along with a brief introduction of BO, PSO, and GA. Section V presents GlobDesOpt's features and the optimization options currently implemented. Section VI shows the optimization results and, finally, conclusions are drawn in Section VII.

II. PROBLEM FORMULATION

In this section, robot kinematic modelling and manipulability measure are presented, along with a brief description of Bayesian Optimization.

A. ROBOT KINEMATIC MODELLING

Different methods exist to describe the kinematic model of robots, one of the most widespread being based on Denavit-Hartenberg (DH) convention [15]. According to this method, the transformation matrix ${}^{i-1}\boldsymbol{T}_i \in \mathbb{R}^{4\times4}$, relating the pose of a link i to the preceding one, can be described by means of 4 parameters $d_{DH,i}, \theta_{DH,i}, a_{DH,i}, \alpha_{DH,i}$ and the joint angle q_i, where $d_{DH,i}$ is the distance between two consecutive joints along the the current joint axis, $\theta_{DH,i}$ the tilting between the two joints about the current joint axis, $a_{DH,i}$ the link's length, and $\alpha_{DH,i}$ the angle between the two joint axis (Figure 1). These parameters can be all stacked together generating the so-called DH table, which consists of as many rows as the number of joints n.

Since the end-effector's pose of a robot is a function of the DH parameters and of the joint values, the DH parameters and the type of joints (which are typically revolute or prismatic) highly affect the performance of the robot in executing a desired task. An optimal choice of these parameters is thus needed to guarantee the best performance.

B. ROBOT MANIPULABILTY

One typical approach to measure the capabilities of a robotic system is the dexterity. There exist various dexterity indices [16] in the literature, but they are mainly derived from the manipulability ellipsoid. An ellipsoid is defined by a core matrix $\boldsymbol{H}$ as $\boldsymbol{z}^T(\boldsymbol{H})^{-1}\boldsymbol{z} \leq 1$ and the singular values of the core matrix $\boldsymbol{H}$ equal the square of the semiaxes' length of the ellipsoid, whereas its determinant can be shown to be proportional to the ellipsoid volume.

For kinematic purposes, it is common to refer to the kinematic manipulability ellipsoid [17], which results by considering the set of joint velocities such that $\dot{\boldsymbol{q}}^T\boldsymbol{q} \leq 1$. By considering that the tip twist is computed as $\boldsymbol{v} = \boldsymbol{J}\dot{\boldsymbol{q}} \in \mathbb{R}^6$, with $\boldsymbol{J}$ the robot Jacobian matrix, it turns out that the kinematic manipulability ellipsoid is deined as:

$$\boldsymbol{v}^T(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}^T(\boldsymbol{q}))^{-1}\boldsymbol{v} \leq 1\,, \tag{1}$$

with $\boldsymbol{H}(\boldsymbol{q}) = \boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}^T(\boldsymbol{q})$. When the robot approaches a singular configuration, the ellipsoid deforms to a line or a point (i.e. null volume ellipsoid), since at least one of the semiaxes reduces to zero. In this work we will be using the standard kinematic manipulability measure:

$$\delta(\boldsymbol{q}) = \sqrt{det(\boldsymbol{H}(\boldsymbol{q}))}\,. \tag{2}$$

Clearly, the robot design affects the dexterity measure by altering the Jacobian matrix.

C. BAYESIAN OPTIMIZATION

Generally, analytically defining a cost function to optimize in terms of the robot design parameters (the DH parameters

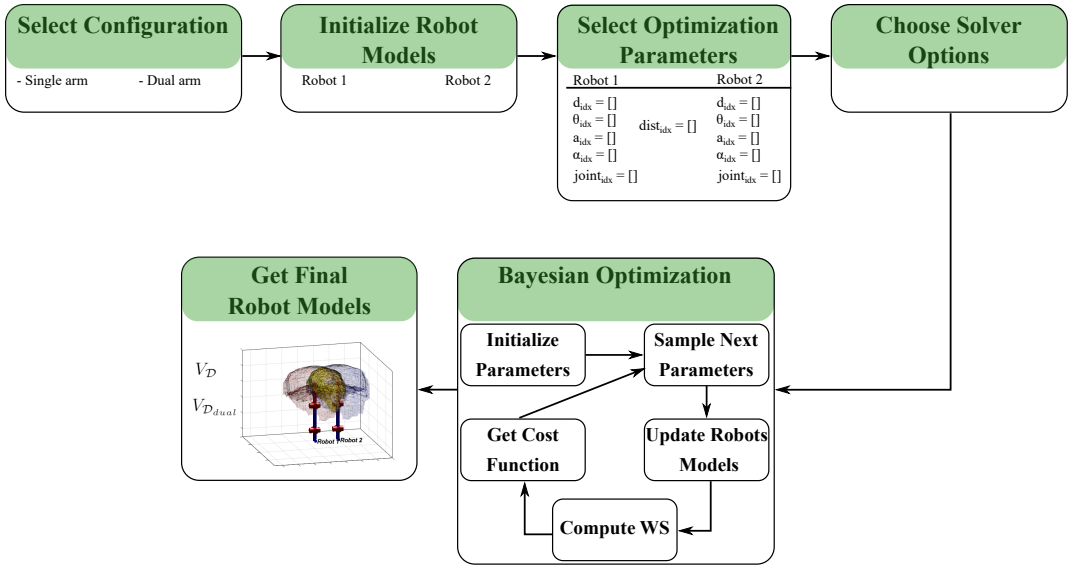


FIGURE 2: GlobDesOpt workflow

and the type of joints) is intractable or even impossible. In this case, the derivatives of the function are unknown, and exploratory techniques need to be employed.

Recently, in the field of machine learning, BO has found great popularity to solve optimization problems with black-box cost functions. Given a cost function $f(x)$, BO attempts to find the global optimum in a minimum number of steps, by incorporating prior belief about $f(x)$ and updates the prior with samples drawn from $f(x)$ to get a posterior that better approximates $f(x)$. Differently from other global optimization methods such as evolutionary algorithms, BO exploration is more probabilistic and exploits the model probability estimate to search through more uncertain regions. The steps performed in the optimization are:

1) **Initialization**: a random number of $NumSeedPoints$ configurations are created and the cost function evaluated.
2) **GP Regression**: a $surrogate\ model$ of $f(x)$ is created based on the sampled points. Generally, a Gaussian Process (GP) regression model is employed. The GP outputs a probability distribution which is used to identify the sampling points.
3) **Sampling**: an $acquisition\ function$ exploits the output of the GP to identify the sampling points, taking into account the exploration and exploitation trade-off to either evaluate points with low mean, or those with high uncertainty, respectively. The new point is added to the current dataset.

Steps 2 and 3 are then repeated until the maximum number of iterations is reached. Typical $acquisition\ functions$ are $Probability\ of\ Improvement$ (PI) [18], $Expected\ Improvement$ (EI) [19] and $Lower\ Confidence\ Bound$ (LCB) [20].

In robot design, generally the numbers of parameters to optimize is not large and BO proved to be very efficient with dimensional problems with up to 20 parameters [12]. However, new approaches to deal with higher dimensional problems are being explored in the literature [21], [22].

III. GENETIC ALGORITHM

GA [23] 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 GA are:

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

Due to its capabilities, GA has been widely used to identify optimal design configurations [24]. Different works employed GA for optimal robotic design [8], [25], [26].

In GA 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 to 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 achieved, or if the best fitness value doesn't change for a certain number of generations ($Stall\ Generations$). At the end of the procedure, the population of the latest generation is returned.

IV. PARTICLE SWARM OPTIMIZATION

Similarly to GA, PSO [27] makes no assumptions about the problem being optimized and can search very large spaces of candidate solutions, and it is derivative-free.

PSO solves the optimization by having a set of candidate solutions (the particles) and moving these particles around in the search-space according their position and velocity. Each particle's movement is influenced by its local best known position, but is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

1) **Initialization**: an initial swarm of particles of $SwarmSize$ size is randomly created individuals. This initial population can be assigned by the user or generated by means of the $CreationFunction$. Similarly, initial random velocities are initialized too.
2) **Function Evaluation**: the objective function at each particle location is computed, determining the best function value and the best location.
3) **Velocity and Location Update**: the velocity of each particle is updated by taking into account the previous velocity, the difference between the current position and the best position the particle has seen, and the difference between the current position and the best position in the current swarm. The particles' positions are then updated with the new velocity.

Steps 2 and 3 are repeated until the stopping criteria are met, e.g if the fitness value is lower than a specific threshold, if the maximum number of generations is achieved, or if the best fitness value doesn't change for a certain number of iterations.

V. GLOBDESOPT FRAMEWORK

In this section, GlobDesOpt is presented, describing its functionalities and workflow.

A. FRAMEWORK DESCRIPTION

GlobDesOpt is an optimization framework to identify the best robot design, given a specific cost function to minimize. GlobDesOpt is meant to be user-friendly and versatile, in order to be used in different scenarios and by inexperienced users. Figure 2 shows the workflow of GlobDesOpt.

Currently, GlobDesOpt allows optimizing for all the DH parameters of each robot's link $d_{DH,i}, \theta_{DH,i}, a_{DH,i}, \alpha_{DH,i}$ and for each joint type (prismatic or revolute). The choice of what parameters to optimize for is left to the user. Additionally, GlobDesOpt supports the following design optimizations:

- single arm optimization: one single kinematic chain is optimized;
- dual-arm optimization: the design of two arms in a dual-arm configuration is simultaneously optimized. In this configuration, the two robots can have the same or completely different kinematic structures. In the former case, one arm is just a copy of the other. In the latter case, the DH parameters and joint types of the two robots can be optimized independently. In both cases, however, the distance between the bases of the two robots $\boldsymbol{d}$ is added as additional optimization variable.

A .xml file is used as configuration file, passing information about the optimization to perform. The path to an initial DH table of the robots can be specified in the .xml file, especially if only some parameters need to be optimized. The robot model is then generated by means of a robot class function, which can also be easily edited by the user according to their individual needs. The user selects what parameters to optimize for, creating a structure of indexes for each parameter in the configuration file. Additionally, lower and upper bounds for the parameters are specified. If the dual arm configuration is chosen, with the two arms having different structures, then the indexes of the parameters for both arms need to be specified too.

Once the optimization variables and their bounds are specified, the solver is called and the optimization performed. Matlab Bayesian Optimization [28], Particle Swarm Optimization [29], and Genetic Algorithm [30] toolboxes are used in this work, and the user can specify different optimization options such as the number of iterations, the size of the population and of the swarm (for PSO and GA), the acquisition function and the exploitation/exploration ratio for BO. As described in II-C, BO runs iteratively and builds a GP model mapping the optimization variables to the cost function. By default Matlab uses ARD Matérn 5/2 kernel for the model. At each step of the optimization, the robots' structures are updated, the dexterous workspace calculated, and the cost function computed. The process is repeated until the maximum number of iterations is reached.

When the optimization finishes, the optimized parameters are returned, along with a plot of the robots' structures and workspaces.

B. SINGLE ARM OPTIMIZATION

Different cost functions can be implemented in GlobDesOpt, however, in this work the goal is to find the optimal design in order to improve the dexterity of a miniaturized robotic structure.

The dexterous workspace (WS) is a subset of the reachable WS, which is the set of Cartesian points that the robot can reach. In order to generate it, each joint is excited sinusoidally, spanning the whole joint's range. Different combinations of frequencies are commanded, in order for the robot to explore as much as possible the whole Cartesian space, as in [16]. The number of samples N_s for generating the reachable workspace is user-defined.

For each configuration $\boldsymbol{q}_{n_s}$, with $n_s = 1 \dots N_s$, the dexterity measure $\delta(\boldsymbol{q})$ defined in (2) is computed, which considers both the position and orientation terms. The user specifies an acceptance rate (α) to select only those configurations, and the corresponding Cartesian points, that have a dexterity measure above the acceptance rate. The dexterous WS is then defined as:

$$\begin{aligned} \mathcal{D}_q &= \{\boldsymbol{q}_{n_s} \mid \delta(\boldsymbol{q}_{n_s}) \geq \alpha(\delta_M - \delta_m) + \delta_m\} \,, \\ \mathcal{D} &= \{\boldsymbol{P}_{n_s} \mid \boldsymbol{q}_{n_s} \in \mathcal{D}_q\} \end{aligned} \tag{3}$$

where $\boldsymbol{P}$ is the robot's tip position, δ_M, δ_m are the maximum and minimum dexterity measures in the explored configurations, $\mathcal{D}_q$ is the set of dexterous joint configurations, and $\mathcal{D}$ a point cloud of dexterous Cartesian points. This point cloud is then converted into a 3D volumetric shape with the $Alphashape$ Matlab function. Once the 3D shape is created, its volume $V_{\mathcal{D}}$ can be easily computed with Matlab's $Volume$ function. As the $Alphashape$ generates a 3D volume of triangular meshes, the $Volume$ function adds up all the volumes of these meshes, returning the total volume of the 3D shape.

In order to take into account both the dexterity of the robot and the size of the dexterous WS for the optimal design, the global dexterity measure can be defined, similarly to [6], as:

$$\Delta = \int_{\mathcal{D}} \delta dV \,. \tag{4}$$

Supposing that the volume is discretized equally into N_v points, then (4) can be computed as:

$$\Delta = \sum_{i=0}^{N_v} \delta_i \frac{V_{\mathcal{D}}}{N_v} = \bar{\delta} V_{\mathcal{D}} \tag{5}$$

and the cost function for the single arm optimization can be defined as:

$$C = -log(\Delta) \,, \tag{6}$$

where the minus sign is needed as Matlab requires a function to minimize and the logarithm is just used to scale the cost.

C. DUAL-ARM OPTIMIZATION

In a dual-arm configuration, the two robots work together to a common task and it is thus important to maximize the common dexterity.

To identify the common dexterous WS, first the two dexterous WS for the two arms $\mathcal{D}_1, \mathcal{D}_2$ as in (3) are computed independently. Then, the the dual arm dexterous workspace is computed as $\mathcal{D}_{dual} = \mathcal{D}_1 \cap \mathcal{D}_2$, which comprises all the points in common. This generates a new point cloud, which is then used to generate a 3D $Alphashape$, for which the volume $V_{\mathcal{D}_{dual}}$ can be computed. Figure 3 displays an example of the evaluation of the common dual-arm WS. Similarly to the single arm optimization, the goal is to maximize both the dexterity of the dual-arm robot and the size of the dexterous WS and, therefore, the global dual-arm dexterity measure is considered:

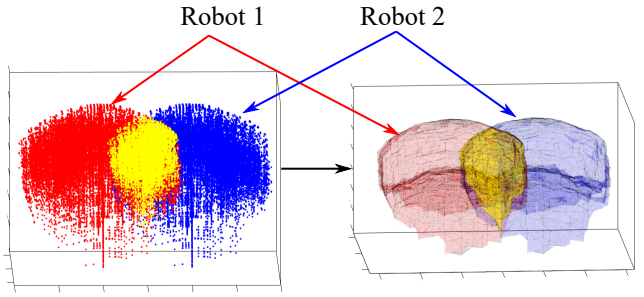


FIGURE 3: Example of the dual-arm dexterous WS generation $\mathcal{D}_1, \mathcal{D}_2$ in point cloud (left) and 3D $Alphashape$ (right) form. The yellow volume is the common dexterous WS $\mathcal{D}_{dual}$.

$$\Delta_{dual} = \int_{\mathcal{D}_{dual}} \delta_{dual} dV \,. \tag{7}$$

Lee at al [5] defined the dual-arm dexterity as the volume of the manipulability ellipsoid resulting from the intersection of the two arms' manipulability ellipsoids. This measure only depends on the joint configurations of the two robots and, in their approach, it is computed locally for specific configurations in a motion task. In order to optimize the robot design and be the least task-specific, it is necessary to compute the dual arm dexterity for different possible combinations of the joint configurations of the two arms, within the common WS.

Therefore, in this work, for each configuration of one of the two arms $\boldsymbol{q}_{1,i} \in \mathcal{D}_{dual}$, the dual-arm dexterity measures $\delta_{dual_{i,j}}$ for all the configurations of the second arm $\boldsymbol{q}_{2,j} \in \mathcal{D}_{dual}$ can be computed. As the goal is to optimize the design in order to maximize the system's dexterity, for each configuration i in the dexterous WS the dual-arm dexterity is computed as $\hat{\delta}_{dual_i} = \min_j(\delta_{dual_{i,j}})$. Finally, the global dual-arm dexterity measure considered in this scenario results to be:

$$\begin{aligned} \Delta_{dual} &= \int_{\mathcal{D}_{dual}} \hat{\delta}_{dual} dV \\ &\simeq \sum_{i=0}^{N_v} \hat{\delta}_{dual_i} \frac{V_{\mathcal{D}_{dual}}}{N_v} \,. \\ &= \bar{\delta}_{dual} V_{\mathcal{D}_{dual}} \end{aligned} \tag{8}$$

Algorithm 1 summarizes the computation of Δ_{dual}.

Based on the current idea, in the case of the dual arm configuration the optimizer would lead the two robots to be as close as possible to each other, since the common volume would then be maximized. However, the smaller the distance between the two robots, the higher the risk of collision.

Algorithm 1 Global dual-arm dexterity measure computation.

function GETGLOBALDUALARMDEXT($\boldsymbol{q}_1$, $\boldsymbol{q}_2$,$\mathcal{D}_{dual}$)
▷ Get volume of common dexterous WS
  $V_{\mathcal{D}_{dual}} = Volume(\mathcal{D}_{dual})$
▷ Get number of configurations of the arms in $\mathcal{D}_{dual}$
  $N_1, N_2 = getConfig(\mathcal{D}_{dual})$
  for $i = 1 \dots N_1$ **do**
    for $j = 1 \dots N_2$ **do**
      $\delta_{dual_{i,j}} = getDualArmDext(\boldsymbol{q}_{1,i}, \boldsymbol{q}_{2,j})$
    end for
  ▷ Compute minimum dual-arm dexterity
    $\hat{\delta}_{dual_i} = \min(\delta_{dual_{i,1}}, \dots, \delta_{dual_{i,N_2}})$
  end for
▷ Compute average dual-arm dexterity
  $\bar{\delta}_{dual} = \frac{1}{N_1} \sum_{i=1}^{N_1} \hat{\delta}_{dual_i}$
▷ Get global dual-arm dexterity measure
  $\Delta_{dual} = \bar{\delta}_{dual} V_{\mathcal{D}_{dual}}$
  return Δ_{dual}
end function

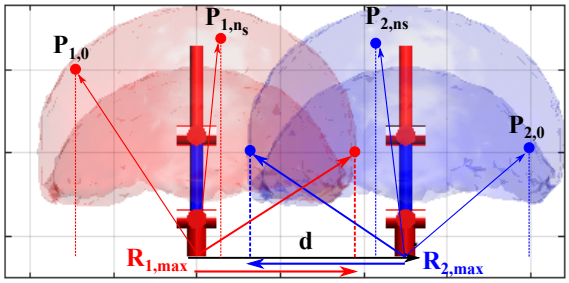


FIGURE 4: The computation of the maximum reach for the dual-arm configuration, given the two arms' reachable WS.

In order to prevent this, a safety measure S is introduced in the cost function for the dual arm. The safety is defined as:

$$S = 1 - e^{-K_1(\frac{||\boldsymbol{d}||}{||\boldsymbol{R}_{max}||})^{K_2}} \tag{9}$$

where $\boldsymbol{d}$ is the distance vector between the two arms' bases, $\boldsymbol{R}_{max}$ is the maximum reach, $K_1 = 7$,$K_2 = 2$ are two tunable parameters. Such function was chosen to have smooth variations in the safety values and such that $\min S = 0$ at low distances and $\max S = 1$ for large distances between the two arms. Note that in the dual-arm configuration, the distance vector $\boldsymbol{d}$ is added as an optimization parameter. The maximum reach is a measure to quantify how far apart the arms can be without ever colliding with each other. Once the reachable WS for each robot is obtained, each Cartesian point is projected onto the distance vector $\boldsymbol{d}$. The maximum reach is computed as:

$$\begin{aligned} &\boldsymbol{R}_{max} = \boldsymbol{R}_{1,max} - \boldsymbol{R}_{2,max} \ , \quad \text{where} \\ &\boldsymbol{R}_{i,max} = \max_{n_s}\left(\frac{\boldsymbol{P}_{i,0}^T \boldsymbol{d}}{||\boldsymbol{d}||^2}\boldsymbol{d}, \dots \frac{\boldsymbol{P}_{i,N_s}^T \boldsymbol{d}}{||\boldsymbol{d}||^2}\boldsymbol{d}\right) , (i = 1, 2) \end{aligned} \tag{10}$$

where $\boldsymbol{P}_{1,n_s}$, $\boldsymbol{P}_{2,n_s}$ $(n_s = 0, \dots, N_s)$ are the points in the reachable WS of the two arms. The minus sign is needed because the reach of the second arm is considered to be in the direction opposite to $\boldsymbol{d}$, when all vectors are expressed in a common reference frame. Figure 4 shows an example to compute $\boldsymbol{R}_{max}$.

From (9), it turns out that dual arm configurations where $||\boldsymbol{d}|| \geq ||\boldsymbol{R}_{max}||$ are the safest, i.e. $S \simeq 1$, as the two arms would never collide. Smaller distances, instead, increase the chance of collisions and, consequently, lower the safety measure. Finally, the cost function for the dual-arm optimization is then defined as:

$$C = -log(S\ \Delta_{dual}) \ , \tag{11}$$

where the safety measure is used as a penalization term to prevent the two arms from being too close to each other.

VI. RESULTS

In this section the design optimization results are shown, presenting a comparison of the optimal design when using the three different solvers for a dual-arm robot, along with the optimization results for a surgical slave robot both in single and dual-arm optimization mode.

A. COMPARISON OF THE OPTIMIZATIONS

In this test GlobDesOpt was employed to optimize the design of a 4-DOF dual-arm robot, with two identical arms. Table 1 reports the initial DH table of the manipulator's arms. The optimization parameters in this scenario are the first and second link lengths l_1, l_2, the joint types of joint 1 and 4, the distance along x, y between the two arms' base frames. The initial robot's configuration is set to be PRRR, meaning the first joint is prismatic and the others are revolute. The

TABLE 1: Initial DH table for each arm of the 4-DOF robot

#	$\mathbf{d}_{\mathbf{DH}}$	$\theta_{\mathbf{DH}}$	$\mathbf{a}_{\mathbf{DH}}$	$\alpha_{\mathbf{DH}}$
1	l_1	0	0	$\pi/2$
2	0	$\pi/2$	l_2	$-\pi/2$
3	0	$-\pi/2$	0	$-\pi/2$
4	5	0	0	$\pi/2$

joint bounds are set to $\begin{bmatrix}0 & 10.0\end{bmatrix}$ mm for the prismatic joints and $\begin{bmatrix}-\pi/4 & \pi/4\end{bmatrix}$ for the revolute ones. The bounds on the link lengths l_1, l_2 are instead set to $\begin{bmatrix}0 & 10.0\end{bmatrix}$ mm and $\begin{bmatrix}3.0 & 25.0\end{bmatrix}$ mm respectively, whereas the bounds on the distance are set to $\begin{bmatrix}0 & 15.0\end{bmatrix}$ mm both for x, y direction.

In this exemplary test only 4 parameters and two joint types are optimized, therefore a brute force space search could be sufficient too, even though the number of computations would depend on the discretization of each parameter. In a more general case, more parameters might need to be optimized and brute force search strategies may be highly computational inefficient. Therefore global optimization approaches might be more beneficial.

All three solvers BO, PSO, and GA are here tested and their results compared. The maximum number of iterations was set to 100 and a population of 50 individuals used

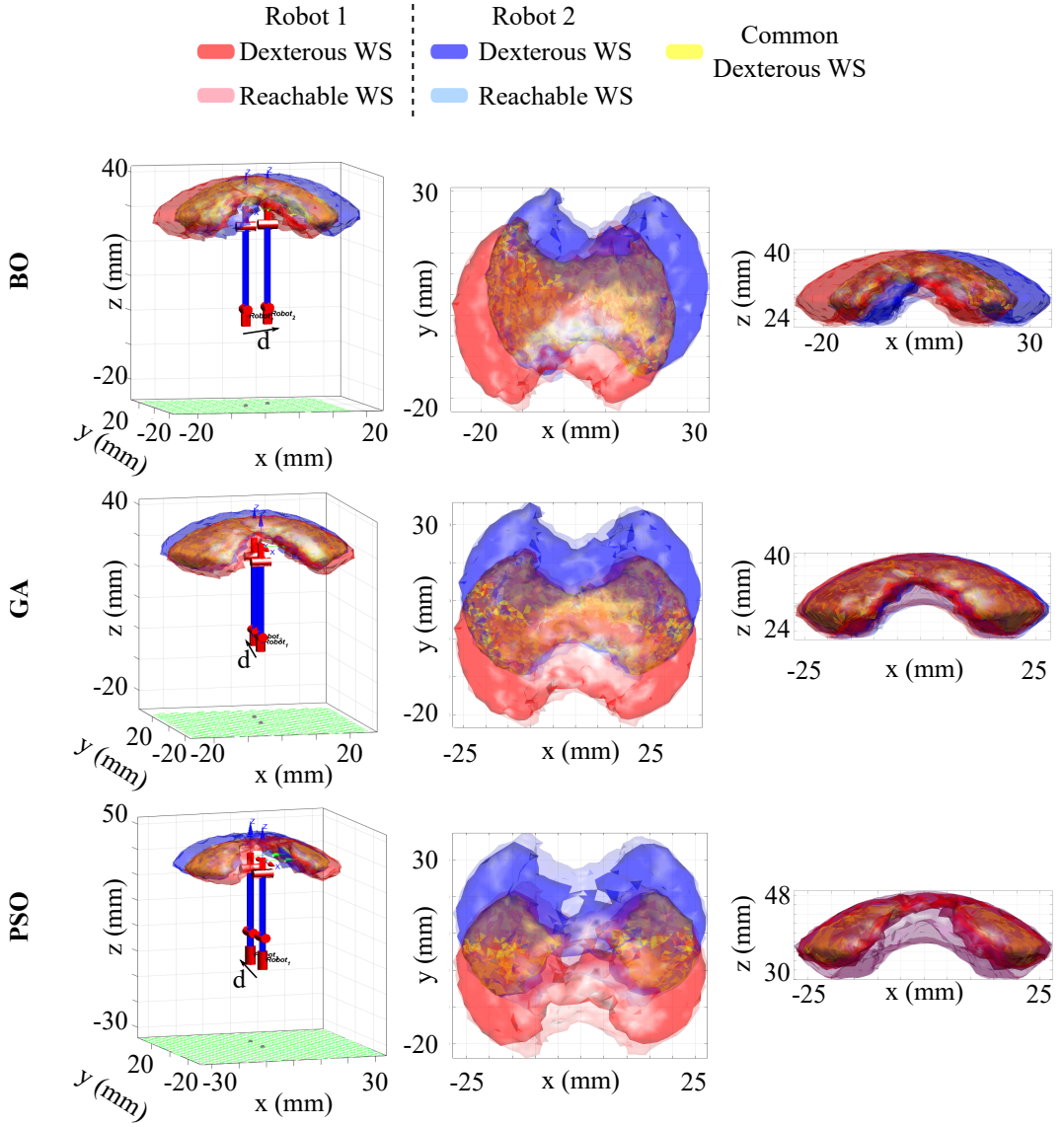


FIGURE 5: Optimization results for the 4-DOF robot using BO,GA,PSO

for GA and for PSO, whereas for BO EI was used as $acquisition\ function$ and 1000 initial points generated. Additionally, the number of stalling iterations both for GA and PSO was set to 10, $50 \cdot 10^3$ points were used to generate the WS in both cases, and $\alpha = 0.6$ is chosen to select the points belonging to the dexterous WS. Table 2 reports the comparison of the performances of the three solvers, and Figure 5 plots the optimized robot design and its reachable and dexterous workspaces.

In all cases, the optimal joint types result to be RRRP, namely the last joint is prismatic. BO and GA also result in same link lengths, but different base offset $\boldsymbol{d}$. An advantage of GA and BO over PSO is that they allow specifying the optimization variables to be integers. This is of great importance because it allows taking into account constraints due to the resolution of the manufacturing process. PSO, instead, would need a proper adjustment or rounding strategy to be implemented. Given the fact that BO and GA produce

TABLE 2: Results for the design optimization of the 4-DOF robot. $V_{\mathcal{D}}$ is the volume of each arms' dexterous WS, $V_{\mathcal{D}_{dual}}$ the volume of the common dexterous WS, $\bar{\delta}_{dual}$ the average dual dexterity in the common dexterous WS, and $log(\cdot) = log_{10}(\cdot)$.

	$\mathbf{l_1}$ (mm)	$\mathbf{l_2}$ (mm)	$\mathbf{d_x}$ (mm)	$\mathbf{d_y}$ (mm)	**Types**	$log(\mathbf{V_{\mathcal{D}}})$	$log(\mathbf{V_{\mathcal{D}_{dual}}})$	$log(\boldsymbol{\delta}_{\mathbf{dual}})$	**Safety**	**Cost**	**Time (s)**
BO	2.0	24.0	8.0	7.0	rrrp	4.0927	3.6702	1.6779	0.2367	-4.7273	129
GA	2.0	24.0	1.0	12.0	rrrp	4.0927	3.7615	1.7048	0.3532	-5.0157	932
PSO	9.09	25.0	0.0	13.18	rrrp	3.9211	3.5096	1.4596	0.3928	-4.5634	852

the same robotic design, the volume of the dexterous WS for the two robot arms is also the same. The volume of the common dexterous WS and the average dual-arm dexterity are however slightly larger for GA optimization, due to the different relative placement of the two arms' bases. This also results in a larger safety value (0.3532 for GA and 0.2367 for BO) and smaller value of the cost function to minimize (-5.0157 and -4.7273 respectively). PSO is the solver that leads to worst performances in terms of volume of the dexterous WS and average dual-arm dexterity.

In terms of computational cost, we used a Windows x64 16 core machine with Intel i9-10980HK CPU and BO results to be the fastest solver: PSO takes 5.6 times longer and GA 6.2 times longer.

B. SURGICAL ROBOT DESIGN OPTIMIZATION

GlobDesOpt has also been employed in this work to optimize the design of a dual-arm surgical robotic instrument. Due to the satisfying and faster performances of BO optimizer, only BO is here employed as solver. The robot consists of two identical arms, whose kinematic model is shown in Figure 6 and the DH parameters in Table 3.

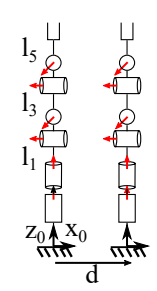


FIGURE 6: The dual-arm surgical robot kinematic model

TABLE 3: DH table for each arm of the dual-arm surgical robot

#	$\mathbf{d_{DH}}$	$\theta_{\mathbf{DH}}$	$\mathbf{a_{DH}}$	$\alpha_{\mathbf{DH}}$
1	0	0	0	0
2	l_1	$\pi/2$	0	$\pi/2$
3	0	$\pi/2$	0	$-\pi/2$
4	0	0	l_3	$\pi/2$
5	0	0	0	$-\pi/2$
6	0	l_5	0	0

Each arm has 6 joints, with the first one being prismatic and the others revolute. The joint limits are set by design constraints to $\left[0, \quad 10.0\right]$ mm for the prismatic joint, $\left[-\pi, \quad \pi\right]$ for the first revolute joint, and $\left[-\pi/4, \quad \pi/4\right]$ for the others. Only the three link lengths l_1, l_3, l_5 are here optimized, whereas the other parameters are set by design considerations.

Even though only the three link lengths are here optimized, standard optimization procedures would still be challenging to implement due to the difficulties in analytically formulating the cost function and its derivatives, especially for the dual-arm configuration.

To evaluate how to optimally design the dual-arm robotic system, two different optimization tests are performed. For both optimizations, EI was used as $acquisition\ function$, $150 \cdot 10^3$ points were used to generate the WS in both cases and $\alpha = 0.7$ is chosen to select the points belonging to the dexterous WS.

1) Independent arm optimization

The design of the arms is optimized without any consideration of the dual-arm configuration. Since the two arms have an identical structure, GlobDesOpt is used with the single arm optimization option and the optimization parameters are set to be the robot's link lengths l_1, l_3, l_5, with their limits set by design considerations to $\left[3.0, \quad 25.0\right]$ mm. In order to keep the robot small enough for surgical applications, the maximum sum of the link lengths is set to be $L_{max} = 47.0$ mm and therefore a linear constraint $l_1 + l_3 + l_5 \leq L_{max}$ is added to the the cost function (6).

2) Simultaneous dual-arm optimization

In this case, the interaction between the two arms is considered and GlobDesOpt is employed in the dual-arm optimization option. In this case, the same optimization parameters as in the single arm case are chosen (l_1, l_3, l_4), yet the distance between the bases of the two arms along the x direction is also added. From design limitations, the bounds for the distance are set to $\left[0.0, \quad 15.0\right]$ mm and, similarly to the independent arm optimization, the constraint on the maximum link lengths is added to the cost function (11).

C. OPTIMIZATION RESULTS

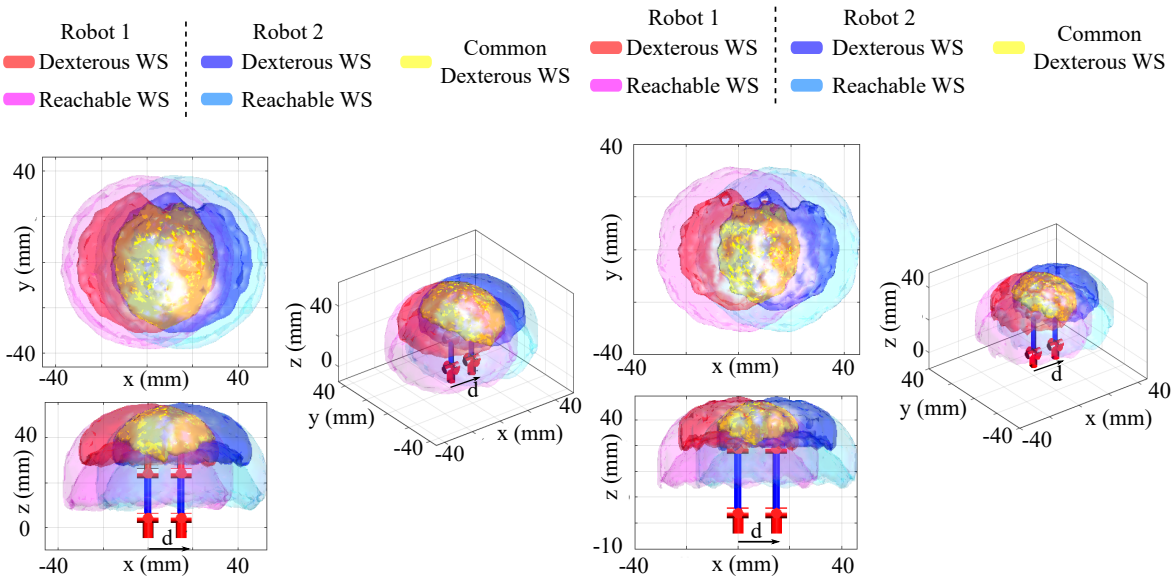
Figure 7 shows the optimized robot designs and Table 4 reports the optimized variables, for the independent and simultaneous optimizations. Since modern manufacturing methods ensure high precision, a resolution of 0.1 mm is allowed. Overall, the independent optimization, which does not consider any interaction between the two arms, resulted in a longer structure, with a total length of 45.6 mm versus 39.1 mm for the simultaneous dual-arm optimization.

To compare the optimal design results of the dual-arm surgical robot, we compared three different cases by positioning the two arms obtained from the independent arm optimization at a distance of 14.8 mm (same as the optimal distance found from the simultaneous dual-arm optimization), 7.4 mm, and 1.5 mm.

Figure 8 shows a comparison of the dexterous WS for the two optimizations in the different placements and Table 5 reports the values of the volumes of the dexterous WS

TABLE 4: Design results for the independent arm and simultaneous dual-arm optimizations. All measures are in mm and N.A. stands for not applicable.

Optimization	$\mathbf{l_1}$	$\mathbf{l_3}$	$\mathbf{l_5}$	**d**
Independent	4.2	19.9	21.5	N.A
Simultaneous	3.8	24.9	10.4	14.8



(a) The independent arm optimization results

(b) The simultaneous dual-arm optimization results

FIGURE 7: The optimal designs and the surgical robot's WS from: 7a) independent arm optimization; 7b) simultaneous dual-arm optimization. In both cases the arms are at a distance of 14.8 mm.

TABLE 5: Comparison of the results from the independent arm and simultaneous dual-arm optimizations for different placements $\boldsymbol{d}$ of the arms Robot 1 and Robot 2. The subscripts $1, 2$ refer to Robot 1 and Robot 2 arms, $V_{\mathcal{D}}$ is the volume of each arms' dexterous WS, $V_{\mathcal{D}_{dual}}$ the volume of the common dexterous WS, $\bar{\delta}_{1,2}$ the arms' average dexterities in their dexterous WS, $\bar{\delta}_{dual}$ the average dual dexterity in the common dexterous WS, and $log(\cdot) = log_{10}(\cdot)$.

Optimization	**d(mm)**	$log(\mathbf{V}_{\mathcal{D}_1})$	$log(\mathbf{V}_{\mathcal{D}_2})$	$log(\mathbf{V}_{\mathcal{D}_{dual}})$	$log(\bar{\delta}_1)$	$log(\bar{\delta}_2)$	$log(\bar{\delta}_{\mathbf{dual}})$	**Safety**	**Cost**
Independent	1.5	**7.5708**	**7.5708**	**7.4747**	4.5836	4.5836	4.2967	$5 \cdot 10^{-5}$	-7.4813
	7.4	**7.5708**	**7.5708**	7.3636	4.5836	4.5836	4.2967	0.0064	-9.4576
	14.8	**7.5708**	**7.5708**	7.1559	4.5836	4.5836	4.2967	0.0500	-10.1515
Simultaneous	14.8	7.2329	7.2329	6.8363	**4.7783**	**4.7783**	**4.4914**	**0.0884**	$\mathbf{-10.2743}$

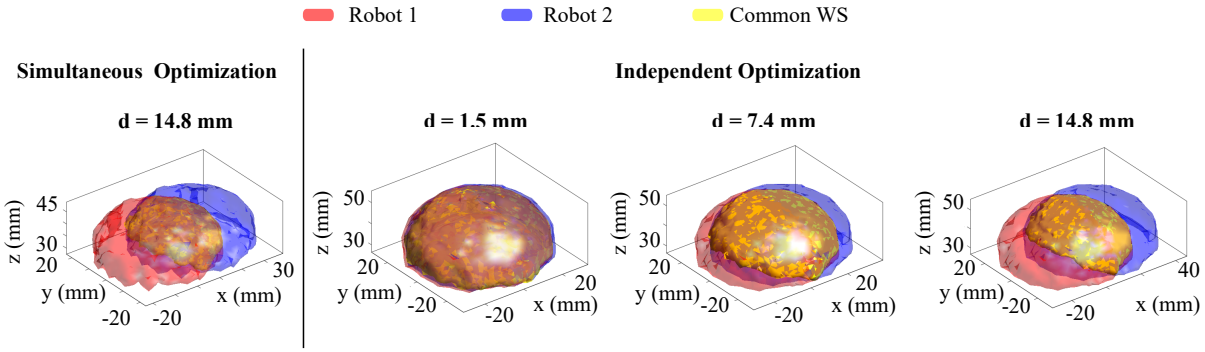


FIGURE 8: Comparison of the dexterous WS from the independent arm and simultaneous dual-arm optimizations for different placements of the arms Robot 1 and Robot 2. The yellow volume is the common dexterous WS $\mathcal{D}_{dual}$.

$\mathbf{V}_{\mathcal{D}_1}, \mathbf{V}_{\mathcal{D}_2}$ for the two arms and of the common dexterous WS $\mathbf{V}_{\mathcal{D}_{dual}}$, the average dexterities of the two arms $\bar{\delta}_1, \bar{\delta}_2$ and the average dual-arm dexterity $\bar{\delta}_{dual}$, the safety value S (9) and the cost function values C (11).

Due to the two arms having the same structure, the volumes of the dexterous WS and the average dexterity measures for the two arms are the same in each test.

In all the three cases from the independent arm optimization, the volume of the dexterous WS does not change, as it is not a function of the distance between the two arms, but just of the arms' Jacobian. The common dexterous WS, instead, is larger when the two arms get closer. Having the two arms too close, however, makes the control more challenging, as the risk of self collisions increases. The proposed safety measure S prevents the arms from being too close and, indeed, it is at its minimum when the distance is set to 1.5 mm. Because of smaller safety values, the cost function values are also larger as the distance decreases.

When comparing with the optimal design results from the simultaneous dual-arm optimization, where the distance is set to be the optimal one of 14.8 mm, the independent optimization leads to higher volumes of the arms' dexterous WS. This is also related to the fact that larger link lengths are obtained with the independent optimization.

The dexterity measures, and especially the dual-arm dexterity measure, result to be higher in the simultaneous dual-arm optimization, as this approach actually considers the interaction between the two arms. Moreover, the safety measure is also larger, leading to a smaller cost.

D. SIMULATED CONTROL TASK

To further validate the best practice to optimally design a dual-arm robot, a motion control test is carried out. The two designs resulting from the independent arm and the simultaneous dual-arm optimizations, with the two arms at a distance of 14.8 mm, are compared.

The control task consists in a rough simulation of a knot tying, with the two arms following a Bernoulli lemniscate path described by:

$$\boldsymbol{P}_{des} = \boldsymbol{P}_c + \begin{bmatrix} \frac{\beta \cos(t)}{1+\sin^2(t)} \\ \frac{\beta \sin(2t)}{2(1+\sin^2(t))} \\ 0 \end{bmatrix}, \qquad (12)$$

where $\boldsymbol{P}_c = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T$ is the center of the path, $t \in \begin{bmatrix} \pi/2, & 3\pi/2 \end{bmatrix}$ for the left arm (Robot 1) and $t \in \begin{bmatrix} -\pi/2, & \pi/2 \end{bmatrix}$ for the right arm (Robot 2).

In order for the path to be within the common dexterous WS and due to the different sizes of the WS resulting from the two optimized designs, the path is scaled for the two tests such that $\boldsymbol{P}_c = \begin{bmatrix} 8.0 & 0.0 & 0.87 z_{max} \end{bmatrix}^T$ mm and $\beta = 0.4 x_{range}$, where z_{max} is the maximum height of the WS (54.0 mm for the independent arm optimization and 46.0 mm for the simultaneous dual-arm optimization) and x_{range} is the span of the WS in the x direction (20.0 mm for the independent arm optimization and 15.0 mm for the simultaneous dual-arm optimization). The two arms are required to follow the two parts of the lemniscate path keeping a constant orientation, in a synchronous and asynchronous motion (Figure 9).

TABLE 6: Dual-arm dexterity measure values in the simulated motion tracking task for the two optimized designs. δ_{ave},δ_{min},δ_{max} are the average, the minimum, and maximum values over the path and $log(\cdot) = log_{10}(\cdot)$.

(a) Synchronous motion

Optimization	$log(\delta_{\mathbf{ave}})$	$log(\delta_{\mathbf{min}})$	$log(\delta_{\mathbf{max}})$
Independent	1.65	-4.06	3.27
Simultaneous	**2.35**	**−1.99**	**3.83**

(b) Asynchronous motion

Optimization	$log(\delta_{\mathbf{ave}})$	$log(\delta_{\mathbf{min}})$	$log(\delta_{\mathbf{max}})$
Independent	-1.14	-2.42	0.21
Simultaneous	**−0.35**	**−1.04**	**0.97**

Results show that the both optimization results achieve good tracking accuracy. However, the simultaneous dual-arm optimization design leads to higher values in terms of dual arm-dexterity throughout the motion (Table 6).

VII. CONCLUSIONS

In this work we presented GlobDesOpt, a simple-to-use open-source Matlab package for optimal robot design, based on three different global optimization solvers. GlobDesOpt has here been employed for two purposes:

- compare the optimization results when using three different global optimization solvers (BO, GA, PSO);
- optimally design a surgical dual-arm robot and compare the results between an independent arm and a simultaneous dual-arm optimization.

From the comparison of the three solvers, it turns out that GA and BO outperform PSO in terms of minimization of the cost function. Moreover GA and BO allow the parameters to be set as integer, thus taking into account numerical approximations due to possible manufacturing resolutions. Among the three, BO results to be the fastest approach. Additionally, results show that for a dual-arm robot, a simultaneous dual-arm optimization should be preferred, as it leads to overall higher dexterity.

Currently GlobDesOpt focuses only on articulated robots whose model is described by DH parameters. However, users can simply adapt to different structures by updating the provided robot class function, including the functions for the forward kinematics and Jacobian computation.

Future work will focus on extending the framework to other robotic structures like parallel or continuum robots, where the design complexity is higher. Moreover, an additional optimization layer will be added in order to optimally identify the best number of joints to consider in the design. We however hope that the community will benefit from this work aimed at simplifying the optimal robot design process and will contribute to use and enhance it.

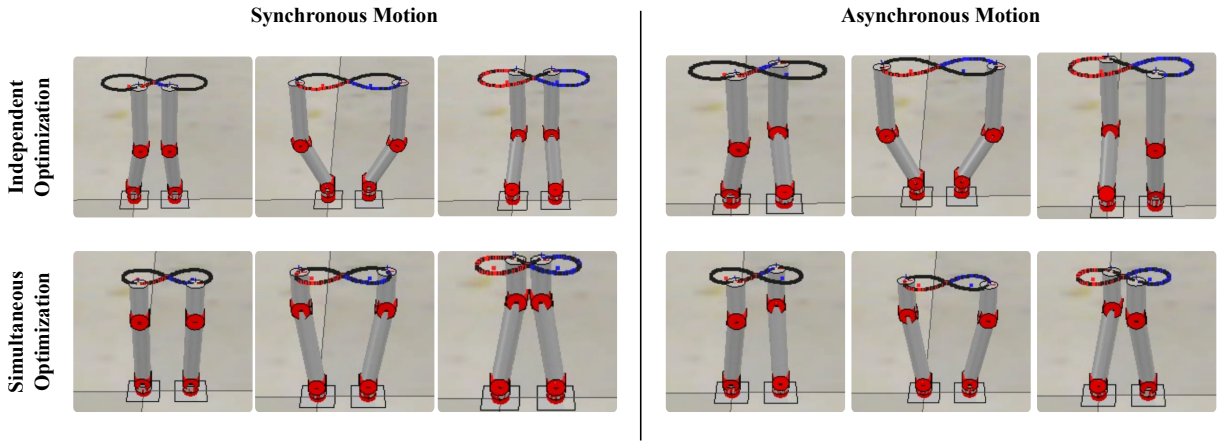


FIGURE 9: Snapshots of the tracking tests for the synchronous and asynchronous motions.

REFERENCES

[1] A. Hassan and M. Abomoharam, "Modeling and design optimization of a robot gripper mechanism," *Robotics and Computer-Integrated Manufacturing*, vol. 46, pp. 94–103, 8 2017.

[2] S. Hwang, H. Kim, Y. Choi, K. Shin, and C. Han, "Design optimization method for 7 DOF robot manipulator using performance indices," *International Journal of Precision Engineering and Manufacturing*, vol. 18, no. 3, pp. 293–299, 3 2017.

[3] J. T. Bryson, X. Jin, and S. K. Agrawal, "Optimal Design of Cable-Driven Manipulators Using Particle Swarm Optimization," *Journal of Mechanisms and Robotics*, vol. 8, no. 4, 8 2016.

[4] C. Baykal, C. Bowen, and R. Alterovitz, "Asymptotically optimal kinematic design of robots using motion planning," *Autonomous Robots*, vol. 43, no. 2, pp. 345–357, 2 2019.

[5] S. Lee, "Dual Redundant Arm Configuration Optimization with Task-Oriented Dual Arm Manipulability," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 1, pp. 78–97, 1989.

[6] A. Torabi, M. Khadem, K. Zareinia, G. R. Sutherland, and M. Tavakoli, "Manipulability of teleoperated surgical robots with application in design of master/slave manipulators," in *2018 International Symposium on Medical Robotics, ISMR 2018*, vol. 2018-January. Institute of Electrical and Electronics Engineers Inc., 4 2018, pp. 1–6.

[7] M. T. Chikhaoui, J. Granna, J. Starke, and J. Burgner-Kahrs, "Toward motion coordination control and design optimization for dual-arm concentric tube continuum robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1793–1800, 7 2018.

[8] A. Schmitz, P. Berthet-Rayne, and G. Z. Yang, "Endoscopic Bi-Manual Robotic Instrument Design Using a Genetic Algorithm," in *IEEE International Conference on Intelligent Robots and Systems*. Institute of Electrical and Electronics Engineers Inc., 11 2019, pp. 2975–2982.

[9] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[10] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, vol. 4, 6 2012, pp. 2951–2959.

[11] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty: An experimental comparison on a dynamic bipedal walker," *Annals of Mathematics and Artificial Intelligence*, vol. 76, no. 1-2, pp. 5–23, 2 2016.

[12] R. Moriconi, M. P. Deisenroth, and K. S. Sesh Kumar, "High-dimensional Bayesian optimization using low-dimensional feature spaces," *Machine Learning*, vol. 109, no. 9-10, pp. 1925–1943, 9 2020.

[13] N. Jaquier, L. Rozo, S. Calinon, and M. B. Urger, "Bayesian Optimization Meets Riemannian Manifolds in Robot Learning," in *Proceedings of the Conference on Robot Learning*. PMLR, 5 2020, pp. 233–246.

[14] X. Zhi, W. Bai, and M. Y. Eric, "Kinematic Parameter Optimization of a Miniaturized Surgical Instrument Based on Dexterous Workspace Determination," in *IEEE International Conference on Advanced Robotics an Mechatronics (ICARM)*, 2021.

[15] B. Siciliano and O. Khatib, *Springer handbook of robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[16] D. Zhang, F. Cursi, and G.-Z. Yang, "WSRender: A Workspace Analysis and Visualization Toolbox for Robotic Manipulator Design and Verification," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3836–3843, 10 2019.

[17] T. Yoshikawa, "Manipulability of Robotic Mechanisms," *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 6 1985.

[18] D. R. Jones, "A Taxonomy of Global Optimization Methods Based on Response Surfaces," *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383, 12 2001.

[19] J. Močkus, "On bayesian methods for seeking the extremum," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 27 LNCS. Springer Verlag, 1975, pp. 400–404.

[20] P. Auer, "Using confidence bounds for exploitation-exploration tradeoffs," in *Journal of Machine Learning Research*, vol. 3, no. 3, 4 2003, pp. 397–422.

[21] M. Mutný and A. Krause, "Efficient High Dimensional Bayesian Optimization with Additivity and Quadrature Fourier Features," Tech. Rep., 2018.

[22] C. Li, S. Gupta, S. Rana, V. Nguyen, S. Venkatesh, and A. Shilton, "High dimensional Bayesian optimization using dropout," in *IJCAI International Joint Conference on Artificial Intelligence*, vol. 0. International Joint Conferences on Artificial Intelligence, 2017, pp. 2096–2102.

[23] D. E. Goldberg and J. H. Holland, "Genetic Algorithms and Machine Learning," pp. 95–99, 1988.

[24] G. Renner and A. Ekárt, "Genetic algorithms in computer aided design," *CAD Computer Aided Design*, vol. 35, no. 8 SPEC., pp. 709–726, 7 2003.

[25] P. K. Jamwal, S. Xie, and K. C. Aw, "Kinematic design optimization of a parallel ankle rehabilitation robot using modified genetic algorithm," *Robotics and Autonomous Systems*, vol. 57, no. 10, pp. 1018–1027, 10 2009.

[26] J. Han, W. K. Chung, Y. Youm, and S. H. Kim, "Task based design of modular robot manipulator using efficient genetic algorithm," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 1. IEEE, 1997, pp. 507–512.

[27] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948.

[28] "Bayesian Optimization Algorithm - MATLAB & Simulink - MathWorks United Kingdom."

[29] "Particle swarm optimization - MATLAB particleswarm - MathWorks United Kingdom."

[30] "Find minimum of function using genetic algorithm - MATLAB ga - MathWorks United Kingdom."