WSRender: A Workspace Analysis and Visualization Toolbox for Robotic Manipulator Design and Verification

Dandan Zhang, *Student Member, IEEE*, Francesco Cursi, *Student Member, IEEE*, and Guang-Zhong Yang, *Fellow, IEEE*

***Abstract*—Workspace analysis is essential for robotic manipulators, which helps researchers study, evaluate, and optimize their designs based on specific criteria with due consideration of ergonomics and usability. Although workspace analysis is a common research topic, current solutions provide design-specific evaluation, and there is a lack of generic software tools for different hardware configurations. This letter presents WSRender, a versatile research-oriented framework for workspace analysis and visualization. It is based on the Orocos Kinematics and Dynamics Library and the Matlab Robotic Toolbox. The software architecture is presented using four use cases for demonstrating its practical use in single robot, dual-arm manipulator performance evaluation, multi-robot interaction analysis, and master–slave mapping. The source code of WSRender[1] is made publicly available for the benefit of the research community for the design or evaluation of robotic manipulators.**

***Index Terms*—Performance evaluation and benchmarking, multi-robot systems, telerobotics, and teleoperation.**

I. Introduction

THE ever-growing deployment of robotic manipulator drives the need for workspace analysis for optimal design and performance evaluation [1]. For example, workspace analysis has been used for design analysis of a flexible snake robot for endoluminal surgery [2], the evaluation of a novel master manipulator [3], and the grasping capability of a robotic arm [4]. Other applications include motion and path planning [5] and optimal task execution strategies [6].

Numerous performance indices have been defined to study the performance and behaviour of manipulators [7], but most of them are tailored to a particular robot design with application-specific configurations [8]. The literature on this topic is rich and detailed surveys can be found in [9]. However, no existing software framework has been developed to support workspace analysis and visualization for various applications using versatile evaluation indices.

This letter presents WSRender, an open workspace analysis and visualization framework, for generic workspace analysis and manipulator design. WSRender provides not only tools to conduct workspace analysis but also visualization and rendering functions, which paves the way for further optimal robotic system design or performance evaluation. The source codes are available for the public, which allows the framework to be extended through additional plugin functions.

WSRender has been tested on a number of robotic platforms for performance assessment, including local evaluation indices generation and global evaluation indices calculation on both kinematic and dynamic aspects. Bimanual manipulation has a wide range of applications from manufacturing [10] to logistics [11], due to its high flexibility and adaptability. Multi-robot cooperative framework has been introduced into the manufacturing workflow [12], while multi-arm surgical robots have also been implemented to enhance the surgical operation efficiency [13]. Master-slave mapping is useful for teleoperation for robotic surgery and space robot control [14]. A general workspace analysis framework, for optimal workspace construction and analysis would therefore be helpful for those research topics.

The contributions of this work can then be summarized as follows:

- development of a general-purpose workspace analysis framework for robotic manipulators;
- introduction of algorithms and data structures for the evaluation of various types of robot;
- incorporation of result visualisation for guiding the robotic system design;
- consideration of different application scenarios;
- provision of extensibility for plug-in and community-based functional extension by users.

The remainder of the letter is structured as follows. Section II illustrates the WSRender software architecture, including the explanation of the overall workflow and key functions of the consecutive procedures. Section III introduces four user cases as examples to verify the usability of the software. Finally, conclusions are drawn in Section IV.

Manuscript received February 24, 2019; accepted June 25, 2019. Date of publication July 22, 2019; date of current version August 1, 2019. This letter was recommended for publication by Associate Editor A. Becker and Editor D. Song upon evaluation of the reviewers' comments. The work of D. Zhang and F. Cursi was supported by the Hamlyn Centre for Robotic Surgery. *(Corresponding author: Dandan Zhang.)*

The authors are with the Hamlyn Centre for Robotic Surgery, Imperial College London , London SW7 2AZ, U.K. (e-mail: d.zhang17@imperial.ac.uk; cursifrancesco@gmail.com; g.z.yang@imperial.ac.uk).

Digital Object Identifier 10.1109/LRA.2019.2929986

[1]The source code of WSRender will be released on https://github.com/HamlynRobotics after this letter is published. (e-mail: d.zhang17@imperial.ac.uk).

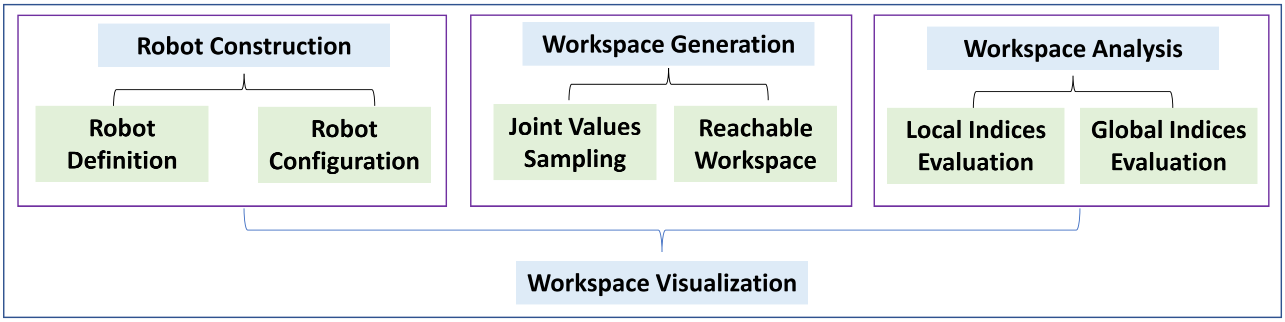

Fig. 1. Software architecture for the WSRender.

II. The WSRender Framework

For workspace analysis, the WSRender provides both Matlab and C++ codes for users, which are based on the Matlab robotic toolbox [15] and the Orocos KDL library [16] respectively. A Matlab based user-friendly Graphical User Interface (GUI) is provided for the designer to tune parameters during the design process, where visualization of the workspace analysis result is implemented for evaluation.

The WSRender framework follows an extensible and modular design protocol. It consists of four major parts (see Fig. 1). These include: i) Robot Construction; ii) Workspace Generation; iii) Workspace Analysis; iv) Workspace Visualization and Rendering. The details about the key components of the WSRender framework and the Matlab GUI are illustrated in the following.

A. Robot Construction

The Robot Construction module consists of robot definition and robot configuration. Robot definition allows building manipulators with specific structure from the Unified Robot Description Format (URDF) file, generated from the Computer Aided Design (CAD) model, or from the Denavit-Hartenberg (DH) table. Robot configuration, instead, determines the placements of the robotic manipulators in the fixed global world coordinate. For scenarios involving multi-robots, the robots can be built in the same world coordinate through pre-defined transformation matrices.

B. Workspace Generation

Given a robotic manipulator model, the goal of workspace generation is to find out the complete set of poses of the manipulator's end-effector in the Cartesian space when the manipulator runs through all possible configurations in the joint space [17]. For workspace generation, analytical [18], geometrical [19], and numerical [20] methods are available. Analytical and geometrical methods, however, may be difficult to apply to complicated robot models. For these reasons, numerical approaches are used in this framework.

A straightforward method for the reachable workspace calculation is to use a combination of joint values sampling and forward kinematics (FK). $\mathcal{W}$ indicates the reachable workspace of the robot, which can be obtained as:

$$\mathcal{W} = \{\mathbf{x} = \mathbf{f}(\mathbf{q}) \mid \mathbf{q_{min}} \leq \mathbf{q} \leq \mathbf{q_{max}}\}\,, \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^3$ is the 3D end-effector position in a Cartesian space, $\mathbf{q} \in \mathbb{R}^n$ is the vector of joint values in the joint space, $\mathbf{q_{min}}$ and $\mathbf{q_{max}}$ represent the lower and upper joint limits respectively, and $\mathbf{f}(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^3$ describes the forward kinematics of the robot.

For numerical calculation, two methods are provided for joint values sampling, which turns the continuous joint values to discrete form. Monte Carlo based technique can be employed to generate a set of combination of random joint values [21]. However, Monte Carlo method cannot guarantee a complete workspace exploration. To cover the entire workspace, the equal interval method can be used, which means discretizing the range of each joint in n_s samples. For a robotic manipulator with n Degrees of Freedom (DoFs), n_s^n possible combination of joint configurations can be obtained [22], which result in n_s^n potential poses of the robot.

For each possible configuration, the corresponding end-effector position in Cartesian space is computed and stored through FK. This will generate a point cloud of the reachable positions in the Catersian space. Once the set of all feasible spatial configurations has been obtained, it is possible to retrieve useful information such as the boundary and volume of $\mathcal{W}$. We use $\mathcal{B}(.)$ to represent the volume of the targeted workspace.

C. Workspace Analysis

For a proper analysis of the performance of a robotic manipulator, computing the reachable workspace is not enough. Performance indices should be calculated after the workspace is generated, which include local and global kinematic and dynamic indices to quantify the kinematic and dynamic behavior of the manipulator respectively. Local indices are posture-dependent, demonstrating a local property; global indices, instead, are posture-independent and are used to identify global workspace characteristics. The computation of the local and global indices as well as the data structures are explained as follows.

1) Local Evaluation Indices: The kinematic and dynamic ellipsoid analysis approaches are popular techniques for local indices evaluation [23], [24]. For the ellipsoid analysis, a core

TABLE I
SUMMARY OF THE WORKSPACE EVALUATION INDICES

	No.(m)	Formulas	Indices	Reference
Local Indices	1/7	$I_{1/7}(\mathbf{q}) = \sqrt{(det(\mathbf{H}(\mathbf{q}))}$	(Kinematic/Dynamic) Manipulability	[24]
	2/8	$I_{2/8}(\boldsymbol{q}) = \sigma_{min}/\sigma_{max}$	(Kinematic/Dynamic) Inverse Condition number	[28] [29]
	3/9	$I_{3/9}(\boldsymbol{q}) = \sigma_{min}(\boldsymbol{q})$	(Kinematic/Dynamic) Minimum Singular Value	[30]
	4/10	$I_{4/10}(\boldsymbol{q}) = \sqrt[n]{(det(\mathbf{H}(\mathbf{q}))}$	(Kinematic/Dynamic) Order-Independent Manipulability	[31]
	5/11	$I_{5/11}(\boldsymbol{q}) = \sqrt{1/tr[(JJ^T)^{-1}]}$	(Kinematic/Dynamic) Harmonic Mean Manipulability Index	[31]
	6/12	$I_{6/12}(\boldsymbol{q}) = \sqrt[n]{(det(\mathbf{H}(\mathbf{q}))}/\bar{\sigma}$	(Kinematic/Dynamic) Isotropic Index	[32]
Global Indices	/	$\tilde{G}_m = \frac{1}{V}\iiint \delta(x,y,z))/\delta_{max} dx\, dy\, dz$	Global Manipulability/Condition Number *etc.*	[33]
	/	$S_l = \frac{1}{\sqrt[3]{V}}\sum_{i=1}^{n}(a_i + d_i)$	Structure Length Index	[34]
	/	$I_s = \sigma_{min}/\sigma_{max}$	Global Isotropy Index	[35]
	/	$\mathcal{B}(\mathcal{W})$	Workspace Volume	

Note: $\sigma_i(i = 1, 2...k)$ are the singular values of $H(\boldsymbol{q})$, $\sigma_{min} = min(\sigma_1, \sigma_2, \ldots, \sigma_k)$, $\bar{\sigma} = mean(\sigma_i)(i = 1, 2...k)$.

matrix can be denoted as $\mathbf{H}(\mathbf{q})$, so the general form of ellipsoid can be defined by

$$\mathbf{P^T}(\mathbf{H}(\mathbf{q}))^{-\mathbf{1}}\mathbf{P} \leq 1. \tag{2}$$

The singular values of the core matrix $\mathbf{H}(\mathbf{q})$ equal the square of the semiaxes' length of the ellipsoid, whereas its determinant can be shown to be proportional to the ellipsoid volume [25]. 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.

For kinematic indices, suppose that $\dot{\mathbf{x}} \in \mathbb{R}^3$ is the end-effector's velocity that is determined by a given joint velocity vector $\dot{\mathbf{q}} \in \mathbb{R}^n$. $\mathbf{J}(\mathbf{q})$ is the Jacobian matrix mapping $\dot{\mathbf{q}}$ to $\dot{\mathbf{x}}$ ($\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$). Considering the set of joint velocities such that $\dot{\mathbf{q}}^{\mathbf{T}}\dot{\mathbf{q}} \leq 1$, then $\mathbf{P} = \dot{\mathbf{x}}$ and $\mathbf{H}(\mathbf{q}) = \mathbf{J}(\mathbf{q})\mathbf{J}(\mathbf{q})^{\mathbf{T}}$.

For dynamic indices, the end-effector acceleration is denoted as $\ddot{\mathbf{x}} \in \mathbb{R}^3$, while the given joint torque input vector is $\boldsymbol{\tau} \in \mathbb{R}^n$. The effect of gravity and external forces cause a shift of the center of the ellipsoid [26]. Considering the set of torques such that $\boldsymbol{\tau}^{\boldsymbol{T}}\boldsymbol{\tau} \leq 1$, then $\mathbf{P} = \ddot{\mathbf{x}}$ and the core matrix can be expressed as $\mathbf{H}(\mathbf{q})^{-\mathbf{1}} = (\mathbf{B}(\mathbf{q})\mathbf{J}^{\dagger}(\mathbf{q}))^{\mathbf{T}}(\mathbf{B}(\mathbf{q})\mathbf{J}^{\dagger}(\mathbf{q}))$, where $\mathbf{B}(\mathbf{q}) \in \mathbb{R}^{nxn}$ is the robot inertia matrix, $\mathbf{J}^{\dagger}$ is the Jacobian pseudoinverse.

Several local kinematic/dynamic evaluations indices based on the core matrix and singular values have been proposed in existing literature and they are summarized in Table I. For all the local indices, normalization is utilized to make the local index invariant to scale, units and reference frame [27]. In this case, the boundary of the local indices is [0, 1].

Beside the kinematic structure, the shape and volume of the workspace are also influenced by the mechanical joint limits of the manipulator, which alters the distribution of local indices. The influence of joint limits can be addressed by adding a penalization term to augment the Jacobian matrix of the robotic manipulator [36]. A more reasonable column-wise penalization term was proposed for both non-redundant robot and redundant robot [20], which is utilized in the current framework. Considering joint redundancy, each column $i = 1...n$ of the Jacobian is penalized through the penalization function such that $\tilde{\mathbf{J}}_{\mathbf{i}}(q_i) = \mathbf{J}_{\mathbf{i}}(q_i)p_i(q_i)$, with

$$p_i(q_i) = \frac{1 - e^{-4K\frac{(q_{max,i}-q_i)(q_i-q_{min,i})}{(q_{max,i}-q_{min.i})^2}}}{1 - e^{-K}}, \tag{3}$$

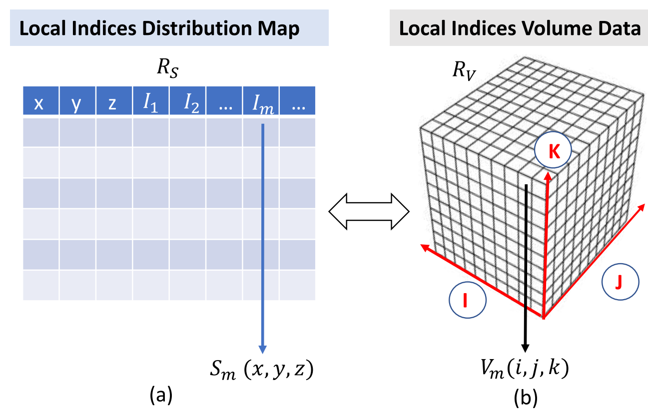

Fig. 2. A schematic illustration of the data structures that are used in the WSRender. (a) Local indices distribution map. (b) Local indices volume data.

where K is a positive constant. This allows the contribution of a specific joint to be zero when it reaches its joint limits.

2) Data Structure: Two forms of data structures are employed in this framework, including the Local Indices Distribution Map, and the Local Indices Volume Data. Fig. 2 demonstrates the relationship between the two data structures.

- **Local Indices Distribution Map**: Suppose that I_m is the m_{th} local evaluation index available in the WSRender, R_S is a space with scatter data, $S_m(x,y,z) \in R_S$ can be generated as the local indices distribution map by (4)

$$S_m(x,y,z) = \mathbf{f}(I_m(\mathbf{q})), \tag{4}$$

where $x_l \leq x \leq x_u, y_l \leq y \leq y_u, z_l \leq z \leq z_u$ (l and u representing the lower and upper boundaries of the point cloud respectively), and $\mathbf{f}(\cdot)$ is the mapping relationship based on FK (see (1)).

- **Local Indices Volume Data**: The workspace can be divided into equally sized cubes called voxels. The x, y and z spaces are divided by the same length of the interval r, which can be known as a predefined precision value. The voxel value can be represented by filling with local indices evaluation information, which describes the capabilities of a robotic manipulator. The corresponding value is mapped to a discrete workspace with cubic voxel, and can be represented as $V_m\,(i,j,k) \in R_V$, where R_V is a space with voxel data.

Algorithm 1: Iterative Method for Global Indices Calculation.

Input: A Pre-defined Robot
Output: Modified Global Evaluation Index $\tilde{G}_m$

Set the iteration number $i = 1$.
Set the initial joint sampling value $m = 15$.
Conduct workspace generation to obtain local indices distribution map based on (4).
Calculate $\tilde{G}_m(1)$.
$i = i + 1$
while $((\tilde{G}_m(i) - \tilde{G}_m(i-1)) > \lambda)$ **do**
    Increase joint sampling number from m to $m + 5$.
    Conduct workspace generation to obtain local indices distribution map based on (4).
    Calculate $\tilde{G}_m(i)$ based on (6).
    $i = i + 1$
end while
$\tilde{G}_m = \tilde{G}_m(i)$

The volume data can be converted by the distribution map, which is described as follows:

$$V_m(i,j,k) = S_m\left(\frac{x - x_l}{r}, \frac{y - y_l}{r}, \frac{z - z_l}{r}\right), \quad (5)$$

where $1 \leq i \leq \frac{(x_u - x_l)}{r}$, $1 \leq j \leq \frac{(y_u - y_l)}{r}$, $1 \leq k \leq \frac{(z_u - z_l)}{r}$.

The two data structures have their respective advantages and disadvantages. The local distribution map consists of a 2D matrix ($n_s^n \times m$), whereas the volume data is a 4D matrix ($\frac{(x_u - x_l)}{r} \times \frac{(y_u - y_l)}{r} \times \frac{(z_u - z_l)}{r} \times m$). Therefore, the former requires less memory storage dimension. Moreover, the volume data has inherent errors, due to the fact that all the scatter points are converted to the centre of the nearest pixel. However, the volume data is more appropriate than the distribution map when conducting multi-robot workspace interaction analysis and master-slave workspace analysis. Using the volume data structure, different robots' workspaces can be easily registered in the same global coordinate system within the same boundary. Since all the data are uniformly distributed across the space occupied by the robots, the interaction calculation can be conducted with ease.

3) Global Evaluation Indices: Most of the local indices distribution across the whole workspace can be exported for further workspace global properties evaluation. A general global modified evaluation index $\tilde{G}_m$ can be regarded as the ratio of the integration of normalized local indices over the whole workspace and the reachable workspace volume, namely:

$$\tilde{G}_m = \frac{1}{V}\iiint \frac{S_m(x,y,z)}{S_m^*(x,y,z)} dx\, dy\, dz\ , \quad (6)$$

where $S_m^*(x,y,z)$ is the maximum value among all $S_m(x,y,z)$, V is the workspace volume. The normalization ensures that it is dimension-independent. This measure can evaluate the robotic manipulator in some average sense [33].

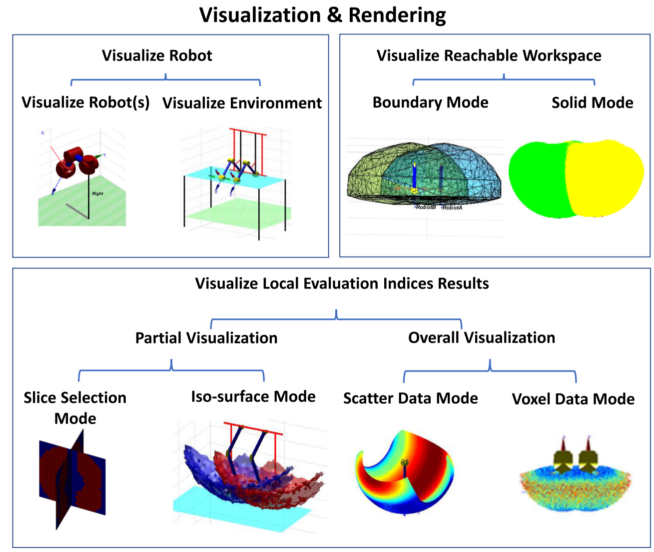

Fig. 3. Visualization and rendering for workspace analysis.

Iterative method can be utilized by defining a tolerable error λ to obtain the global evaluation value when it converges. The iterative calculation is summarized in Algorithm 1.

Other global evaluation indices include the global isotropy index, structure length index etc. Global isotropy index is the ratio of the minimum to the maximum singular values for the entire workspace. Structural length index ($S_l = \frac{1}{\sqrt[3]{V}}\sum_{i=1}^{n}(a_i + d_i)$) is based on the ratio of the sum of all the link lengths and joint offsets of the manipulator to the cube root of the workspace volume [34], where a_i and d_i represent the link length and link offset of the i_{th} joint respectively, n is the number of joints for the robot, and V is the workspace volume. The mathematical definition of the available local and global evaluation indices are summarized in Table I.

D. Visualization and Rendering

The WSRender Workspace Visualization module includes three parts, i.e. i) robot and environment visualization, ii) workspace generation results visualization and iii) workspace analysis results visualization. Fig. 3 shows the examples of visualization effects provided by the WSRender.

For robot visualization, a single robot or dual-arm manipulator can be built and visualized, while simple environment can be constructed to show the relative position of the robots in the global frame. Visualization of workspace generation results can show the reachable workspace through boundary mode or solid mode. α is a parameter that controls the transparency of the workspace boundary during visualization.

For workspace analysis results visualization, partial visualization allows the user to obtain the local indices distribution value at a selected slice S when slice selection mode is used. With the iso-surface mode, the high quality workspace area can be separated from the low quality area based on a pre-defined threshould value μ. As for overall visualization, the local indices distribution across the whole reachable workspace are visualized. The

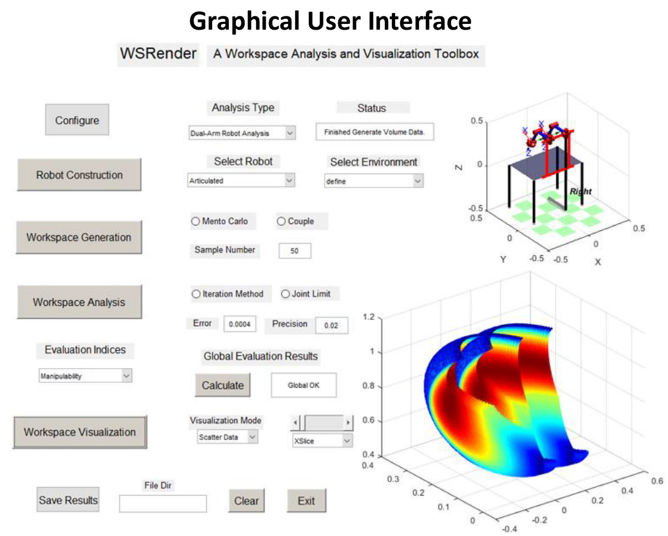

Fig. 4. Overview of the Matlab GUI.

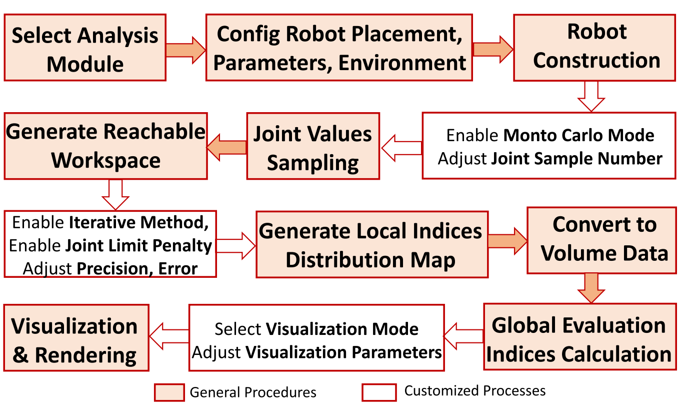

Fig. 5. Workflow illustration for the usage of WSRender.

results can be presented in scatter data mode and volume data mode, which represents the local indices distribution map and the local indices volume data.

E. *Overview of the User Interface*

A Matlab GUI (Fig. 4) is designed to simplify the workspace analysis and visualization process. The analysis type can be selected at first, including single robot analysis, dual-arm manipulator analysis, multi-robot interaction analysis, and master-slave mapping analysis.

For single or bimanual manipulator analysis, all the procedures can be conducted using the GUI directly. As for the multi-robot interaction and master-slave mapping analysis, different types of the robots should be built separately. A set of C++ codes is also provided to build the robots' workspaces and compute the local indices distribution map. Then, the computed map of the robots can be loaded into the Matlab GUI for visualization and further analysis.

The standard workflow of the usage of the GUI is illustrated in the schematic sketch (see Fig. 5). After selecting the analysis module and configuring the robot (setting its placement, the key parameters for workspace generation and analysis, and the environment setup), the robot is constructed and visualized in the specified environment. Subsequently, the joint sampling method can be chosen for the workspace and local indices computation. The local indices distribution map is then converted to volume data for global indices calculation or further multi-robot interaction analysis. Finally, the workspace is visualized and rendered.

TABLE II
PARAMETERS FOR WSRENDER

		Definition	Default Value
Mode	/	Monto Carlo Method	Disable
	/	Iterative Method	Disable
	/	Joint Limit Penralization	Disable
Calculation	r	Precision	$\sqrt[3]{V}/100$
	λ	Tolerable Error	0.0001
	n_s	Joint Number	15
Visualization	α	Alpha Value	0.1
	μ	Threshold	0.9
	S	Slice	X = 0, Y = 0, Z = 0

Different calculation modes can be selected and several key parameters can be fine tuned. Table II indicates the default values for the computation modes and key parameters for the WSRender. Users can modify the values based on their application requirements [37], [38]. Changes can also be easily reverted by saving and reloading the key parameters.

III. USE CASES

This section gives four use cases of WSRender to verify its effectiveness and demonstrate its usability. These include performance evaluation for single manipulator and dual-arm manipulator, workspace analysis for the multi-robot system, and master-slave mapping.

A. *Single Manipulator Evaluation*

Several typical models are included in the WSRender, whose library can be enlarged by easily integrating the URDF files or DH table of the existing robots. For example, the Master Tool Manipulators (MTMs) of the da Vinci Robotic System [39], a general-purpose haptic interface Phantom Omni (Geomagic Touch) [40], Puma560 [41] and Stanford Arm [42] are included in the library of WSRender.

MTMs and Omni can be used as master manipulators for medical robotics, where the kinematic performance is a key aspect of evaluation. Puma560 [41] and Stanford Arm [42] are typical manipulators for industrial application, whose dynamic properties are significant.

Table III shows the global kinematic evaluation results of the Phantom Omni and the da Vinci MTMs and global dynamic evaluation results of the Puma560 and Stanf as examples for single robotic manipulator global kinematic/dynamic workspace evaluation.

B. *Dual-Arm Manipulator Evaluation*

In this case study, two comparable robots are used for workspace analysis, i.e. a spherical robot and an articulated robot. The spherical robot has a prismatic joint and five revolute

TABLE III
GLOBAL EVALUATION RESULTS

Indices	Omni	MTM	Indices	Puma560	Stanf
$\tilde{G}_1$	0.553	0.558	$\tilde{G}_7$	0.053	0.061
$\tilde{G}_2$	0.524	0.466	$\tilde{G}_8$	0.029	0.070
$\tilde{G}_3$	0.525	0.476	$\tilde{G}_9$	0.036	0.072
$\tilde{G}_4$	0.855	0.868	$\tilde{G}_{10}$	0.605	0.619
$\tilde{G}_5$	0.273	0.524	$\tilde{G}_{11}$	0.634	0.592
$\tilde{G}_6$	0.860	0.860	$\tilde{G}_{12}$	0.655	0.722

TABLE IV
DH TABLE AND JOINT LIMITS FOR A SPHERICAL ROBOT AN ARTICULATED ROBOT FOR CASE STUDY

		1	2	3	4	5	6
Spherical Robot	α	0	90°	-90°	0	-90°	90°
	a	0	0	0	0	0	0
	d	0	0	d_3	0	0	0
	θ	θ_1	θ_2	0	θ_4	θ_5	θ_6
	θ_u	45°	135°	0.40m	180°	160°	70°
	θ_l	-45°	45°	0.20m	-180°	20°	-70°
Articulated Robot	α	0	90°	0	90°	-90°	90°
	a	0	0	0.20m	0.20m	0	0
	d	0	0	0	0	0	0
	θ	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
	θ_u	135°	45°	0	180°	160°	70°
	θ_l	45°	$-45°$	-120°	-180°	20°	-70°

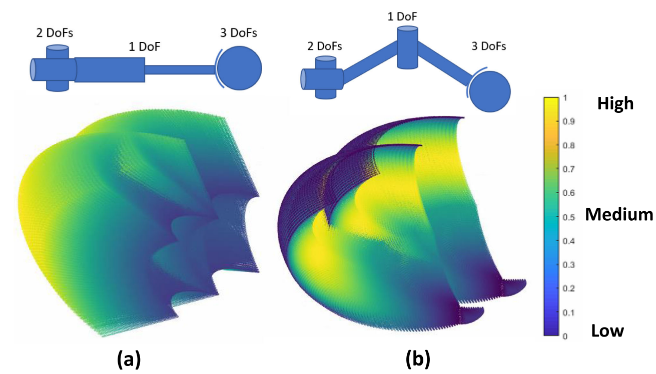

Fig. 6. Comparison of the distribution of the kinematic manipulability index (I_1) between (a) a dual-arm spherical robot, (b) a dual-arm articulated robot.

joints, while the articulated robot has six revolute joints. The modified DH table and their upper and lower joint limits can be found in Table IV, where $\theta_i (i = 1, 2, \ldots, 6)$ and d_3 indicate joint variables.

After generating the local indices distribution map, visualization and rendering tool is used to enable the designer to be aware of the workspace quality by having an intuitive sense of the local indices distribution. Comparison of the distribution of the kinematic manipulability index between the spherical robot and the articulated robot can be viewed in Fig. 6.

C. Multi-Robot Interaction Calculation

WSRender provides a module for multi-robot interaction workspace analysis. The boundary of the volume data is determined to cover the reachable workspace of the targeted robots in a pre-defined environment. Suppose that there are K robots in

TABLE V
PLACEMENT OF THE ROBOTS

		X	Y	Z	RX	RY	RZ
Spherical Robot	Right	0.1m	0.0m	0.4m	0°	0°	0°
	Left	-0.1m	0.0m	0.4m	0°	0°	0°
Articulated Robot	Right	0.1m	0.4m	0.0m	0°	0°	90°
	Left	-0.1m	0.4m	0.0m	0°	0°	90°

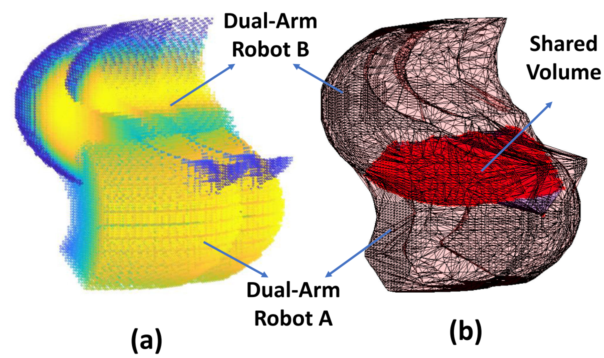

Fig. 7. The overall workspace and the shared workspace determination and analysis of a multi-robot system. (a) Multi-robot overall workspace visualization. (b) Generation and visualization of the shared workspace among multi-robots.

total for analysis. After calculating the local indices distribution map of each robot, the local indices volume data for each robot can be converted. For the reachable workspace representation, if a voxel is reachable by a robot, it is assigned the value of 1, otherwise, it remains 0.

In this use case, a dual-arm spherical robot (Robot A) and a dual-arm articulated robot (Robot B) form a multi-robot system. The two robots have the same DH parameters and joint limits in Table IV. The placements of the robots are defined in Table V, where $P = [X, Y, Z]$ is the position transform vector, while RX, RY, and RZ are the Euler angles generated by the rotation matrix along the axis x, y and z of the global frame respectively.

As shown in Fig. 7(a), the reachable workspace of all the robots are generated respectively and visualized. Let $V_p(i, j, k)$ be the reachable workspace volume data of the p_{th} robot, whose entries are either 0 or 1. The overall volume data of the multi-robot system is V, which is computed as:

$$V(i, j, k) = \sum_{p=1}^{p=K} V_p \left(i - r_x^p, j - r_y^p, k - r_z^p\right), \qquad (7)$$

where $R_p = (r_x^p, r_y^p, r_z^p)$ is the translation vector of the frame of the p_{th} robotic manipulator to the global frame. V_s is the set of all the voxels with value of 2 among all the voxels in V, while V_a is the set of all the voxels with the value larger than 0 among all the voxels in V. The volumes $\mathcal{B}(V_s)$ and $\mathcal{B}(V_a)$ can be computed.

The percentage of the volume of the shared workspace to the overall reachable workspace generated by the multi-robots can be defined as $R = \frac{\mathcal{B}(V_s)}{\mathcal{B}(V_a)}$. The values of $\mathcal{B}(V_a)$, $\mathcal{B}(V_s)$ and R are shown in Table VI, which are generated by numerical calculation.

TABLE VI
MULTI-ROBOT INTERACTION ANALYSIS RESULTS

	$\mathcal{B}(V_a)$	$\mathcal{B}(V_s)$	R
Dual-Arm Robot A	$0.065\ m^3$	$0.020\ m^3$	0.31
Dual-Arm Robot B	$0.061\ m^3$	$0.014\ m^3$	0.23
Multi-Robot System	$0.110\ m^3$	$0.017\ m^3$	0.15

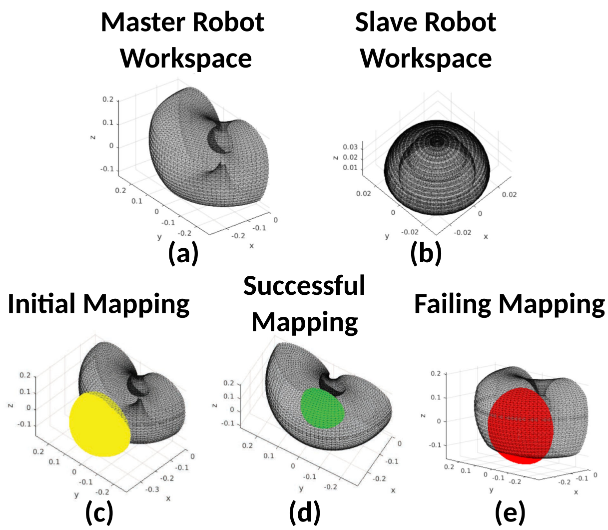

Fig. 8. A use case scenario for master-slave mapping workspace analysis. All units are in [m]. (a) Master robot workspace visualization. (b) Slave robot workspace visualization. (c) Initial master-slave mapping demonstration. (d) Successful master-slave mapping demonstration. (e) Failing master-slave mapping demonstration.

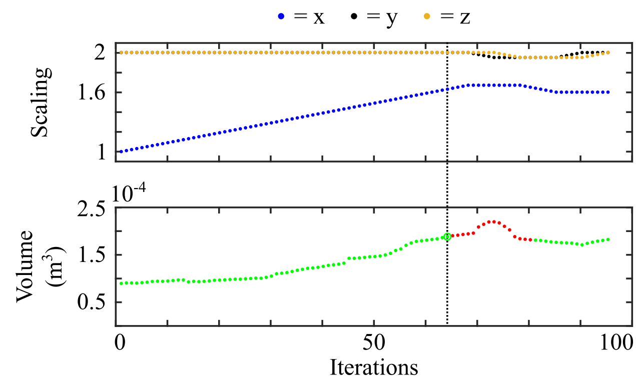

Fig. 9. Optimal volume selection vs scaling, for a given offset of $[-0.2\ 0\ 0]$ m.

D. Master-Slave Mapping Evaluation

Another important aspect, especially for tasks such as telemanipulation is the mapping between the master and the slave robots' workspaces [3]. Because of the usual dissimilarities of the two workspaces, motion scaling and clutching are often required. Clutching is needed when the master robot reaches its workspace boundary and fails to further control the slave robot. In this case, repositioning of the master robot is required while keeping the slave robot still. Clutching can be avoided when the master manipulator's workspace can cover the slave robot's workspace completely after being scaled down by a ratio. In order to support the decision of choosing a proper scaling ratio and offset for the master and slave robot's workspace, an extended module is provided for master-slave mapping analysis.

In this use case, a 5 DoFs articulated robot with full revolute joints is selected as the slave robot, while the master robot is a 4 DoFs robot consisting of a revolute base joint and 3 DoFs parallel arm. The master and the slave workspaces can be generated in their respective reference frames at first (see Fig. 8(a) and (b)).

The scaling factor can be independently adjusted along the three different directions, allowing the slave workspace to elongate or shrink with respect to the slave frame. The offset is defined as the distance vector between the origins of the slave reference frame and of the master. An initial user-defined offset and scaling can be assigned. Fig. 8(c) shows an example of the initial master-slave mapping relationship with scaling of $[5\ 5\ 5]$ and offset $[-0.2\ 0\ 0]$ m. While the user performs the scaling and offsets regulation, the volume of the scaled slave workspace is computed. In order to compute the volume, the point-cloud workspace is converted to an *Alphashape*. Then, Matlab's *volume* function is used to retrieve the volume value.

A continuous check is performed to identify whether the slave robot's workspace is within the master manipulator's workspace or not. Whenever at least one point of the slave workspace exits the master workspace, it can be concluded that the master and slave workspace are mismatched. In case of failure, both the result window and the slave workspace turn red (Fig. 8(e)), otherwise they are green (Fig. 8(d)). An optimal mapping relationship can be obtained by adjusting the offset and scaling ratio. The optimal scaling (the one allowing to have the largest slave volume) found so far is saved and shown. Fig. 9 illustrates an example about how the volume changes as a function of the scaling (for a given offset) and how the optimal scaling is retained.

IV. CONCLUSIONS

This work presents WSRender, an open framework for workspace analysis and visualization, with the goal of making research in this field more efficient by providing a collection of basic algorithms and letting researchers focus on actual tasks.

The capability of this software is verified on different robotic platforms for different application purposes, such as single robot or dual-arm manipulator analysis, multi-robot interaction, and master-slave optimal mapping. The user interface allows prototyping new algorithms or tuning parameters with ease.

Moreover, all components in WSRender are written in a flexible and extensible way, which allows user-specific definition. New evaluation indices can be easily added to the WSRender in the Matlab codes by contributing columns of the workspace evaluation table. Similarly, to allow flexibility of modeling different kinds of robotic systems, the C++ code consists of a single class with virtual functions for kinematics and dynamics computation. This allows the workspace analysis to be customized. The master manipulator analyzed in this letter is a closed-chain robot, which is an example to prove the flexibility of the framework.

All the algorithms, models, codes documentation, data and other example files required for the use cases are available online. We would like to encourage the robotic research community to take advantage of this research-friendly software.

Acknowledgment

The authors would like to thank S. Treratanakulchai for providing a CAD model for case study, and G. Gras and K. Leibrandt for valuable discussions.

References

[1] J. S. Dai, D. Wang, and L. Cui, "Orientation and workspace analysis of the multifingered metamorphic hand—metahand," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 942–947, Aug. 2009.

[2] N. Patel *et al.*, "Evaluation of a novel flexible snake robot for endoluminal surgery," *Surg. Endoscopy*, vol. 29, no. 11, pp. 3349–3355, 2015.

[3] P. Wisanuvej *et al.*, "Master manipulator designed for highly articulated robotic instruments in single access surgery," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 209–214.

[4] I. Akinola, J. Varley, B. Chen, and P. K. Allen, "Workspace aware online grasp planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 2917–2924.

[5] K. Shengzheng, H. Wu, L. Yao, and D. Li, "Coordinated workspace analysis and trajectory planning of redundant dual-arm robot," in *Proc. 13th Int. Conf. Ubiquitous Robots Ambient Intell.*, 2016, pp. 178–183.

[6] K. Leibrandt, C. Bergeles, and G.-Z. Yang, "Implicit active constraints for safe and effective guidance of unstable concentric tube robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 1157–1163.

[7] S. Patel and T. Sobh, "Manipulator performance measures-a comprehensive literature survey," *J. Intell. Robot. Syst.*, vol. 77, no. 3–4, pp. 547–570, 2015.

[8] K. Abdel-Malek, J. Yang, D. Blackmore, and K. JOY, "Swept volumes: Fundation, perspectives, and applications," *Int. J. Shape Model.*, vol. 12, no. 1, pp. 87–127, 2006.

[9] J. Snyman, L. du Plessis, and J. Duffy, "An optimization approach to the determination of the boundaries of manipulator workspaces," *J. Mech. Des.*, vol. 122, no. 4, pp. 447–456, 2000.

[10] Y. Hu, L. Zhang, W. Li, and G.-Z. Yang, "Robotic sewing and knot tying for personalized stent graft manufacturing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 754–760.

[11] R. Kapadia, S. Staszak, L. Jian, and K. Goldberg, "Echobot: Facilitating data collection for robot learning with the amazon echo," in *Proc. 13th IEEE Conf. Autom. Sci. Eng.*, 2017, pp. 159–165.

[12] B. Huang, M. Ye, Y. Hu, A. Vandini, S.-L. Lee, and G.-Z. Yang, "A multirobot cooperation framework for sewing personalized stent grafts," *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1776–1785, Apr. 2018.

[13] F. Zhang, Z. Yan, and Z. Du, "Preoperative planning for the multi-arm surgical robot using PSO-GP-based performance optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 4208–4214.

[14] S. C. Low and L. Phee, "A review of master–slave robotic systems for surgery," *Int. J. Humanoid Robot.*, vol. 3, no. 4, pp. 547–567, 2006.

[15] P. I. Corke, *Robotics, Vision & Control: Fundamental Algorithms in MATLAB*, 2nd ed. New York, NY, USA: Springer-Verlag, 2017.

[16] H. Bruyninckx, "Open robot control software: the OROCOS project," in *Proc. ICRA. IEEE Int. Conf. Robot. Automat. (Cat. No. 01CH37164)*, vol. 3, 2001, pp. 2523–2528.

[17] O. Bohigas, M. Manubens, and L. Ros, "A complete method for workspace boundary determination on general structure manipulators," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 993–1006, Oct. 2012.

[18] Y. Tsai and A. Soni, "Accessible region and synthesis of robot arms," *ASME J. Mech Des*, vol. 103, pp. 803–811, 1981.

[19] A. K. Dash, I.-M. Chen, S. H. Yeo, and G. Yang, "Workspace generation and planning singularity-free path for parallel manipulators," *Mechanism Mach. Theory*, vol. 40, no. 7, pp. 776–805, 2005.

[20] K. Leibrandt *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.

[21] A. N. Chaudhury and A. Ghosal, "Workspace of multifingered hands using Monte Carlo method," *J. Mechanisms Robot.*, vol. 10, no. 4, 2018, Art. no. 041003.

[22] C. Mummolo, F. Cursi, and J. H. Kim, "Balanced and falling states for biped systems: Applications to robotic versus human walking stability," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2016, pp. 1155–1160.

[23] T. Yoshikawa, "Manipulability of robotic mechanisms," *Int. J. Robot. Res.*, vol. 4, no. 2, pp. 3–9, 1985.

[24] T. Yoshikawa, "Dynamic manipulability of robot manipulators," *Trans. Soc. Instrum. Control Eng.*, vol. 21, no. 9, pp. 970–975, 1985.

[25] J.-P. Laumond *et al.*, *Robot Motion Planning and Control*. New York, NY, USA: Springer-Verlag, 1998, vol. 229.

[26] P. Chiaacchio and M. Concilio, "The dynamic manipulability ellipsoid for redundant manipulators," in *Proc. IEEE Int. Conf. Robot. Autom. (Cat. No. 98CH36146)*, 1995, vol. 1, pp. 95–100.

[27] J. Lenarčič and F. Thomas, *Advances in Robot Kinematics: Theory and Applications*. New York, NY, USA: Springer-Verlag, 2013.

[28] O. Ma and J. Angeles, "The concept of dynamic isotropy and its applications to inverse kinematics and trajectory planning," in *Proc., IEEE Int. Conf. Robot. Autom.*, 1990, pp. 481–486.

[29] O. Ma and J. Angeles, "Optimum design of manipulators under dynamic isotropy conditions," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1993, pp. 470–475.

[30] V. Klema and A. Laub, "The singular value decomposition: Its computation and some applications," *IEEE Trans. Autom. Control*, vol. AC-25, no. 2, pp. 164–176, Apr. 1980.

[31] J.-O. Kim and K. Khosla, "Dexterity measures for design and control of manipulators," in *Proc. IROS'91; IEEE/RSJ Int. Workshop Intell. Robots Syst.' 91*, 1991, pp. 758–763.

[32] R. Hashimoto, "Harmonic mean type manipulatability index of robotic arms," *Trans. Soc. Instrum. Control Eng.*, vol. 21, no. 12, pp. 1351–1353, 1985.

[33] C. Gosselin and J. Angeles, "A global performance index for the kinematic optimization of robotic manipulators," *J. Mech. Des.*, vol. 113, no. 3, pp. 220–226, 1991.

[34] S. Kucuk and Z. Bingul, "Comparative study of performance indices for fundamental robot manipulators," *Robot. Auton. Syst.*, vol. 54, no. 7, pp. 567–573, 2006.

[35] L. J. Stocco, S. E. Salcudean, and F. Sassani, "Optimal kinematic design of a haptic pen," *IEEE/ASME Trans. Mechatronics*, vol. 6, no. 3, pp. 210–220, Sep. 2001.

[36] T. Yoshikawa, *Foundations of Robotics: Analysis and Control*. Cambridge, MA, USA: MIT Press, 1990.

[37] T. K. Tanev, "Kinematics of a hybrid (parallel–serial) robot manipulator," *Mechanism Mach. Theory*, vol. 35, no. 9, pp. 1183–1196, 2000.

[38] L. Cui and J. S. Dai, "Posture, workspace, and manipulability of the metamorphic multifingered hand with an articulated palm," *J. Mechanisms Robot.*, vol. 3 , no. 2, 2011, Art. no. 021001.

[39] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. DiMaio, "An open-source research kit for the da vinci surgical system," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 6434–6439.

[40] T. H. Massie *et al.*, "The phantom haptic interface: A device for probing virtual objects," in *Proc. ASME Winter Annu. Meeting, Symp. Haptic Interfaces virtual Environ. Teleoperator Syst.*, 1994, vol. 55, no. 1, pp. 295–300.

[41] A. A. Jaber, "Puma 560 robot and its dynamic characteristics," in *Design of an Intelligent Embedded System for Condition Monitoring of an Industrial Robot*. New York, NY, USA: Springer-Verlag, 2017, pp. 75–91.

[42] T. J. Tarn, A. K. Bejczy, A. Isidori, and Y. Chen, "Nonlinear feedback in robot arm control," in *Proc. 23rd IEEE Conf. Decis. Control*, 1984, pp. 736–751.