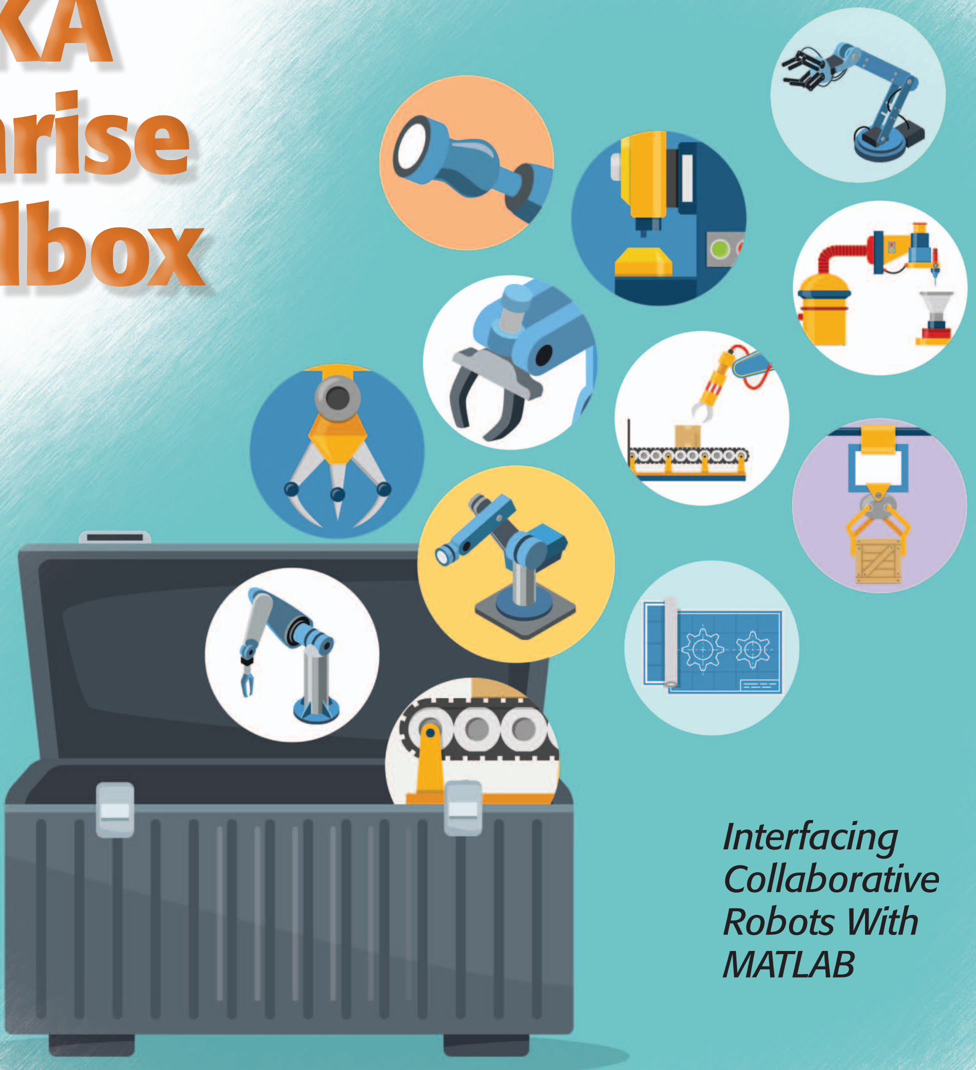
KUKA Sunrise Toolbox



Interfacing Collaborative Robots With MATLAB

TOOLBOX—©ISTOCKPHOTO.COM/JEMASTOCK, ROBOTIC CIRCLES—©ISTOCKPHOTO.COM/ALFAZETCHRONICLES

By Mohammad Safeea and Pedro Neto

Collaborative robots are increasingly present in our lives. The KUKA LBR iiwa, equipped with the KUKA Sunrise.OS controller, is one example of a collaborative/sensitive robot. This tutorial presents the KUKA Sunrise Toolbox (KST), a MATLAB toolbox that interfaces with KUKA Sunrise.OS. KST contains functionalities for networking, soft control in real time, point-to-point motion, parameter setters/getters, general purpose, and physical interaction. It includes approximately 100 functions and runs on a remote computer connected with the KUKA Sunrise controller via Transmission Control Protocol/Internet Protocol (TCP/IP). The potentialities of the KST are demonstrated in nine application examples.

Digital Object Identifier 10.1109/MRA.2018.2877776
Date of publication: 20 November 2018

A Motivation and Related Work

Collaborative robots have been extensively studied and are increasingly safe, intuitive to use, and robust. KUKA Sunrise.OS equips the KUKA LBR iiwa and the mobile platform

Table 1. The iiwa_stack (ROS) and KST interfaces.

	iiwa_stack	KST
Operating system	Linux	Linux, Windows, Mac
Programming languages	C++ or Python	MATLAB
Required knowledge	ROS	MATLAB
3-dimensional simulator	gazebo	v-rep

KMR iiwa. However, because of its functionalities (advanced programming, planning, and configuration), it is expected that the Sunrise.OS will be available for other KUKA robots in the future. These robots are being used in several projects for advanced applications, such as the ColRobot, EURECA, and CogIMon projects [1], [2], with possible implementations in other domains, as demonstrated by the KUKA innovation awards. In these applications, the robot works side by side with the coworker, providing assistance in an intuitive and safe manner.

In the robotics field, several MATLAB toolboxes have been introduced. One of the most popular is the Robotics Toolbox for MATLAB [3]. This toolbox includes functionalities for robotic manipulators, such as homogeneous transformations, forward and inverse kinematics, forward and inverse dynamics, and trajectory generation. The Dynamics Simulation Toolbox for industrial robot manipulators can be used to simulate robot dynamics in addition to other functionalities [4]. The DAMAROB Toolbox allows kinematic and dynamic modeling of manipulators [5]. The KUKA control toolbox is dedicated to the motion control of KUKA manipulators equipped with the KUKA robot controller [6]. JOpenShowVar is a Java-based open-source platform used to interface KUKA industrial robots equipped with the controller KRC4 and KRC2 [7].

Collaborative robots have been extensively studied and are increasingly safe, intuitive to use, and robust.

Recently, KUKA launched the LBR iiwa series of manipulators [8], a commercial version of the KUKA-DLR lightweight robot [9]. The LBR iiwa is a lightweight, sensitive robot with seven axes (redundant). Each axis contains multiple sensors allowing position and impedance control. These robots are programmed with KUKA Sunrise.Workbench, the programming environment for KUKA Sunrise. Input–output (I/O) connectors and an EtherCAT interface are available in the robot flange.

From an external computer, the user can interface with Sunrise.OS using the Fast Robot Interface (FRI) [9] and Robot Operating System (ROS) [10]. The FRI is a platform for controlling the KUKA iiwa remotely from a personal computer (PC), allowing hard, real-time control at rates of up to 1 kHz. ROS is popular within the research community because it allows users to use an external PC to interface with robotic systems. The iiwa_stack [11] is one of the most commonly used packages for interfacing Sunrise.OS with ROS. This package is built on the SmartServo interface only; as such, it provides soft, real-time control capability, allowing the user to control iiwa with the ROS on the fly. As an alternative, KST is the unique MATLAB-based interface for Sunrise.OS (Table 1), and it is a plug-and-play solution that requires no special configuration in the controller and covers a wide range of Sunrise.OS features, especially for human–robot interaction capabilities.

Compared with the iiwa_stack, KST also includes extra functionalities for hand-guiding, precision hand-guiding, nonblocking motion calls, conditional motion calls, and point-to-point motion calls (arcs, lines, and ellipses), among others. In addition, KST provides numerous examples, ranging from simple tutorials on the straightforward implementation of its functions to more complex examples that involve robot control based on external hardware/sensor inputs. KST can be used in education, research, and industry. Thanks to its MATLAB interface, KST facilitates the development of applications for KUKA Sunrise.OS. Moreover, it makes the development of robot applications accessible to people with basic MATLAB skills, even if they are not programming experts.

The KUKA Sunrise.OS controller is programmed using Java, which allows implementation of complex algorithms in the robot controller. Due to communication delay, the use of an external computer to interface with the robot has a negative impact on real-time execution of the robot's commands. In KST, tests revealed a 3- to 4-ms communication delay when sending commands to and receiving commands from the robot, as detailed in the extended version of this tutorial in the supplemental multimedia material. In this scenario, the use of an external computer is advantageous in several cases:

1) when interfacing with multiple external devices
2) when easy integration of external software modules and hardware devices is needed
3) when complex algorithms (image processing and machine learning, among others) requiring high computational power are being implemented
4) when the number of computations involved is relatively high so that performance is limited by the robot controller hardware (unlike the robot controller, the computer hardware can be easily upgraded).

Original Contribution

KST contains more than 100 functions and presents multiple advantages.

1) Easy and fast interaction with the robot is possible from an external computer running KST, which is an open-source solution provided under Massachusetts Institute of Technology license.
2) KST integrates diverse functionalities including, among others, kinematics, motion definition, and precision hand-guiding [the latter refers to robot hand-guiding at

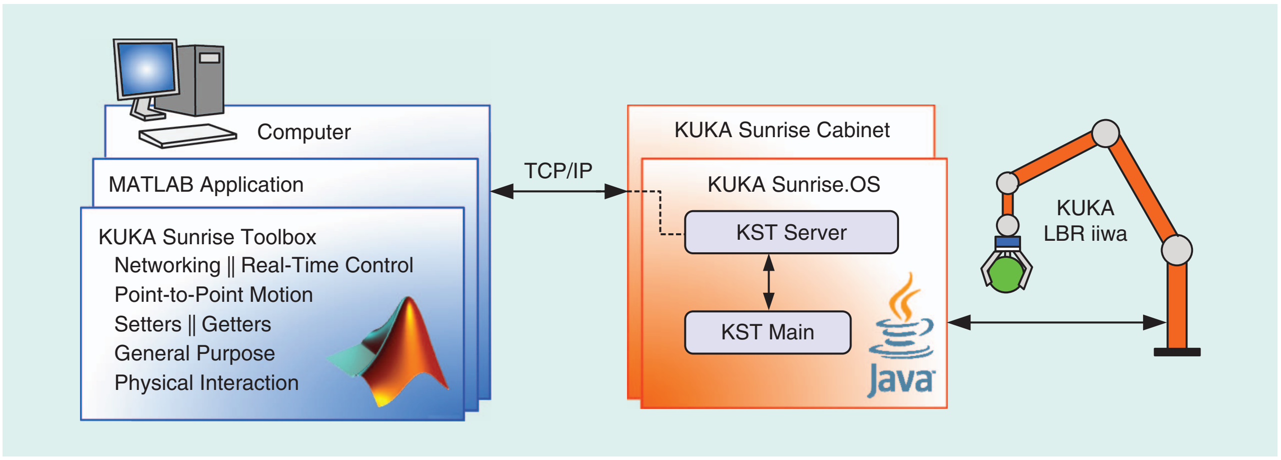


Figure 1. The KST architecture and communication scheme. The KUKA Sunrise cabinet is the physical robot controller.

the end-effector level (translations and rotations for precision positioning) [12], which is different from KUKA's off-the-shelf hand-guiding at the joint level].

3) The toolbox can accelerate the development of advanced robot applications in MATLAB. Complex algorithms and advanced mathematical tools supported by MATLAB (e.g., matrix operations and data filtering) can be implemented in an external computer. Existing software modules/toolboxes (vision, machine learning, statistics, etc.) can also be integrated, extending applications with new functionalities.
4) KST makes the KUKA LBR iiwa manipulators accessible to a wide variety of people from different backgrounds and opens the door of collaborative robotics to many potential new users for academic, educational, and industrial applications. MATLAB is widely used to accelerate the process of research implementation.
5) KST runs inside MATLAB, so it can be used on different operating systems: Windows, Linux, and macOS.

KST

KST functions are divided into seven categories:

1) *Networking:* establishes (and terminates) connection with the robot controller.
2) *Soft real-time control:* activates/deactivates soft real-time functionalities for motion and impedance control. In such a case, the robot's path can be updated online from external sensor data with simultaneous control–feedback capability. These functionalities are built on the DirectServo and SmartServo from KUKA.
3) *Point-to-point motion:* allows point-to-point motion in joint space as well as in Cartesian space. This category also includes nonblocking and conditional/interruptible motion functions.
4) *Setters:* set parameter values in the robot controller (e.g., robot poses, light-emitting diodes, and I/O connectors).
5) *Getters:* obtain parameter values from the robot controller (joint angles, end-effector position, end-effector orientation, force/moment acting on the end effector, joint torques, I/O connectors).
6) *General purpose:* calculates forward and inverse kinematics, the mass matrix, the Coriolis matrix, and the Jacobian matrix.
7) *Physical interaction:* activates/deactivates hand-guiding, precision hand-guiding [12], and double tap detection.

KST runs on an external/remote computer and communicates with KUKA Sunrise via TCP/IP through an Ethernet network using the robot's X66 connector (Figure 1). KST implements a TCP/IP client that communicates with the Java server (KST Server and KST Main) running on Sunrise.OS. Both KUKA iiwa manipulators (KUKA iiwa 7 R800 and KUKA iiwa 14 R820) are supported.

In these applications, the robot works side by side with the coworker, providing assistance in an intuitive and safe manner.

The main functions of KST are detailed and illustrated with implementation examples in the extended version of this tutorial in the multimedia material available in IEEE *Xplore.* The KST toolbox can be freely downloaded from the GitHub repository at https://github.com/Modi1987/KST-Kuka-Sunrise-Toolbox provided under MIT license.

Application Examples

The robot, with a pen mounted on the flange, draws a circle on the top of a white box. This task is achieved by KST point-to-point functionalities that support arc motion. An illustrative example/algorithm is given in Algorithm 1.

This task can also be achieved with the inverse kinematics solver and the soft, real-time control functionalities. The end-effector circular path is calculated while the robot is moving. The inverse kinematics solver is used to calculate the joint angles of the robot from the circular path. The calculated angles are

Algorithm 1: Drawing a circle (MATLAB Code)

```
% Instantiate the KST object
ip = '172.31.1.147'; % IP of the robot controller
arg1 = KST.LBR7R800; % Robot type/model
arg2 = KST.Medien_Flansch_elektrisch; % Flange type/model
Tef_flange = eye(4); % End-effector to flange transform
iiwa=KST(ip,arg1,arg2,Tef_flange);
% Connect to the robot controller
iiwa.net_establishConnection();
% Define circle radius and robot velocity
r = 50; vel = 150;
% Define the center of the circle as the current end-effector position
Cen = iiwa.getEEFPos();
% Define sPoint as the starting point of the circle
sPoint = Cen; sPoint{1} = sPoint{1}+r;
% Move the end-effector to the starting point of the circle
iiwa.movePTPLineEEF(sPoint,vel);
% Specify the parameters of the arc, namely the angle theta
% subtended by the arc at the center of the rotation and the
% XY coordinate of the center of the arc
theta = -2*pi;
c = [Cen{1}; Cen{2}];
% Move the end-effector to perform the arc motion
iiwa.movePTPArcXY_AC(theta,c,vel);
```

relayed to the Sunrise controller through KST such that the robot end effector executes the circular path. Figure 2(a) illustrates the control scheme flowchart, and Figure 2(b) is a snapshot of the robot drawing a circle. Because robot motion is being generated online using KST's soft, real-time control functionalities the path can quickly be adjusted to change velocity or stop the motion, if desired. The example code can be found in the MATLAB file KSTclass_Tutorial_drawCircle.m.

Easy and fast interaction with the robot is possible from an external computer running KST.

KST was also used to implement a practical use case in which a human coworker and the robot share workspace and subtasks related to the assembly of two parts joined by screws. As such, KST was used to implement a human–robot collision avoidance system based on the well-known potential fields method [13]. The screwing operation (Figure 3) for a single screw is divided into three subtasks.

First, the human coworker approaches the work piece to place the screw into the hole while the robot moves away to avoid collision. In this phase the coworker rotates the screw (1–2 turns) and leaves the area. The collision avoidance system is able to adjust the offline preplanned paths smoothly and on the fly to avoid collisions with the dynamic coworker [Figure 3(a)–(c)].

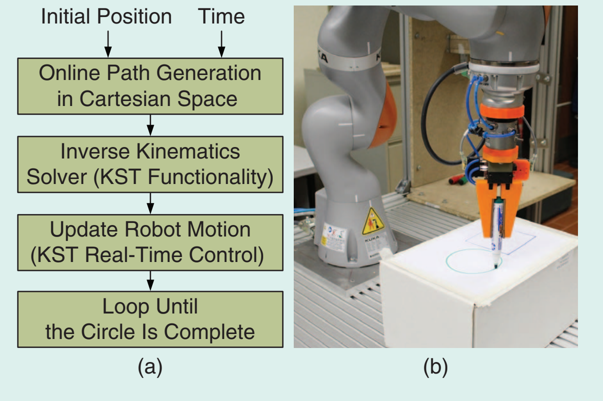

Initial Position
Time
Online Path Generation in Cartesian Space
Inverse Kinematics Solver (KST Functionality)
Update Robot Motion (KST Real-Time Control)
Loop Until the Circle Is Complete
(a)
(b)

Figure 2. Example 1. (a) A flowchart of the MATLAB script. (b) The robot drawing a circle.

Next, the robot automatically returns to the preplanned path to tighten the screw. It approaches the screw head from the top, while the tool attached to the robot end effector starts rotating. When a given torque is reached, the tool stops rotating, and the robot moves up. In this phase, when the robot reaches a predefined distance until it moves up from the screw head, collision avoidance is deactivated. In this scenario, the robot velocity is relatively reduced, limiting the risk for the human coworker [Figure 3(d)–(f)].

Finally, the human approaches the work piece to apply a final manual tightening with adequate pressure.

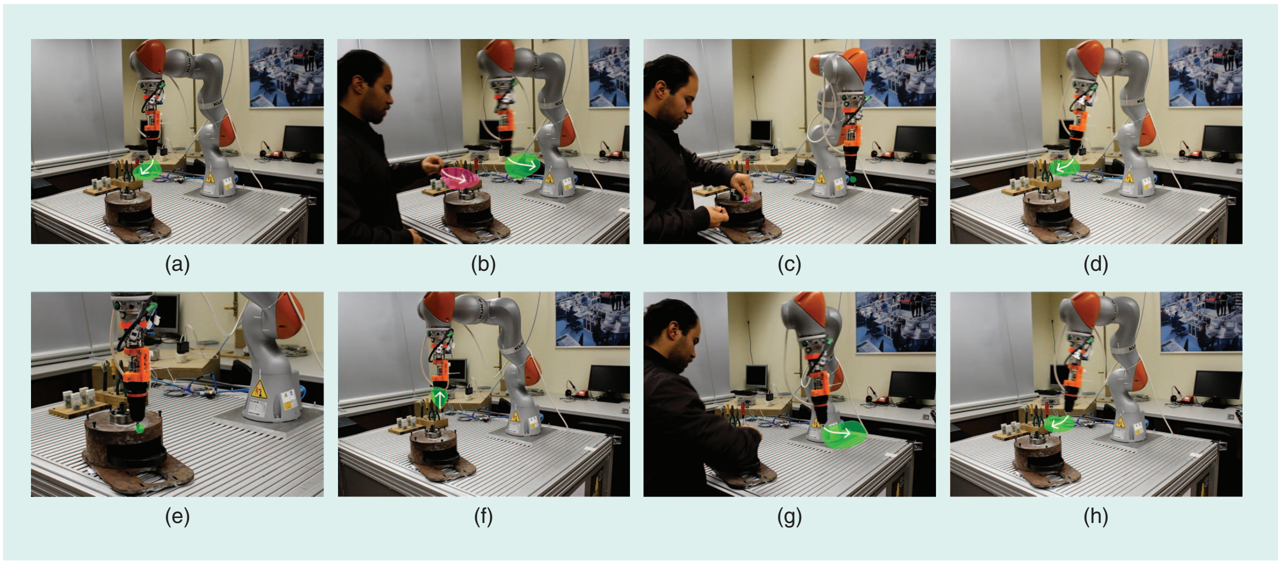


Figure 3. (a)–(h) The robot smoothly avoids collision when the human coworker approaches to perform the screwing operation.

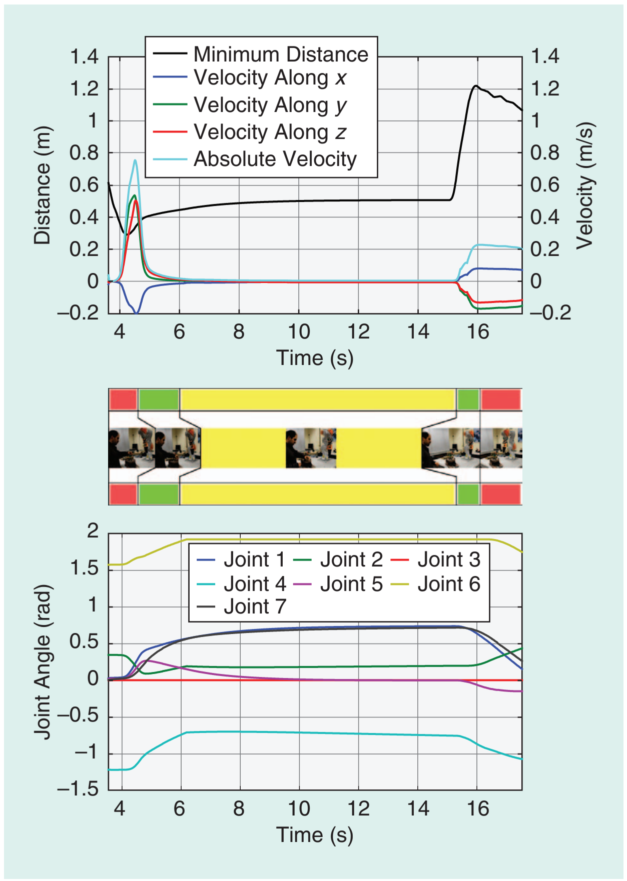


Figure 4. The human–robot minimum distance, robot velocity, and joint angles recorded during experimental tests.

The robot moves away to avoid collision [Figure 3(g) and (h)].

In this system, a magnetic tracker captures the human's pose around the robot. Each magnetic tag provides position and orientation data, which are used as inputs to compute the minimum distance between the human and the robot, both geometrically approximated by capsules. The robot motion is controlled with the soft, real-time control functions provided by KST. We conducted a quantitative analysis by recording the human–robot minimum distance, robot velocity, and robot joint angles (Figure 4). This analysis focused on the first subtask and the beginning of the second subtask [Figure 3(a)–(e)]. At the start, the robot is stationary. When the human coworker approaches the work piece, the human–robot minimum distance decreases to 0.3 m (minimum), and the robot reacts to avoid collision (Figure 4). After this process, when the coworker is placing the screw, the minimum distance is stable, and the robot is stopped, keeping a given safe distance. When the human moves away (second subtask), the robot returns back to the work piece. The sample video and another collision avoidance example with additional detail are provided in the extended version of this tutorial in the multimedia material in IEEE *Xplore*.

The collision avoidance system is able to adjust the offline preplanned paths smoothly and on the fly to avoid collisions with the dynamic coworker.

Nine application examples on a KUKA iiwa 7 R800 manipulator demonstrate the performance and easy use of KST for drawing geometries, DirectServo control, human–robot collision avoidance, teleoperation, hand-guiding and teaching, interfacing Sunrise with v-rep, and controlling iiwa using a graphical user interface. The application examples also include two practical use cases, one for assembly operation using screws and the other for pick-and-place operation. These examples are detailed in the extended version of this tutorial in the multimedia material in IEEE *Xplore* .

KST functionalities are advantageous for the implementation of advanced robot applications.

Conclusions

According to users' feedback (students, researchers, and industry engineers), the proposed toolbox is a useful and intuitive tool to interface with KUKA Sunrise.OS and, in particular, to speed up the development and implementation of robot applications. KST functionalities are advantageous for the implementation of advanced robot applications. KST also facilitates integration of external hardware, data processing, and implementation of complex algorithms using existing toolboxes.

Acknowledgment

This research was partially supported by the Portugal 2020 project DM4Manufacturing POCI-01-0145-FEDER-016418, UE/FEDER through the program COMPETE2020, and the Portuguese Foundation for Science and Technology SFRH/BD/131091/2017.

References

[1] L. Roveda, S. Haghshenas, A. Prini, T. Dinon, N. Pedrocchi, F. Braghin, and L. M. Tosatti, "Fuzzy impedance control for enhancing capabilities of humans in onerous tasks execution," in *2018 IEEE 15th Int. Conf. Ubiquitous Robots (UR)*, pp. 406–411.

[2] S. S. M. Salehian and A. Billard, "A dynamical-system-based approach for controlling robotic manipulators during noncontact/contact transitions," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 2738–2745, Oct. 2018. doi: 10.1109/LRA.2018.2833142.

[3] P. I. Corke, "A robotics toolbox for MATLAB," *IEEE Robot. Autom. Mag.*, vol. 3, no. 1, pp. 24–32, Mar. 1996. doi: 10.1109/100.486658.

[4] M. Toz and S. Kucuk, "Dynamics simulation toolbox for industrial robot manipulators," *Computer Applications Eng. Educ.*, vol. 18, no. 2, pp. 319–330, 2010. doi: 10.1002/cae.20262.

[5] M. Bellicoso. (2010). DAMAROB Toolbox. [Online]. Available: http://www.damarob.altervista.org

[6] F. Chinello, S. Scheggi, F. Morbidi, and D. Prattichizzo, "KUKA control toolbox," *IEEE Robot. Autom. Mag.*, vol. 18, no. 4, pp. 69–79, Dec. 2011. doi: 10.1109/MRA.2011.942120.

[7] F. Sanfilippo, L. I. Hatledal, H. Zhang, M. Fago, and K. Y. Pettersen, "Controlling KUKA industrial robots: flexible communication interface JOpenShowVar," *IEEE Robot. Autom. Mag.*, vol. 22, no. 4, pp. 96–109, Dec. 2015. doi: 10.1109/MRA.2015.2482839.

[8] KUKA. (2018). LBR iiwa. [Online]. Available: https://www.kuka.com/en-my/products/robotics-systems/industrial-robots/lbr-iiwa

[9] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schaeffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger, "The KUKA-DLR lightweight robot arm—a new reference platform for robotics research and manufacturing," in *ISR 2010 (41st Int. Symp. On Robotics) and ROBOTIK 2010 (6th German Conf. on Robotics)*, June 2010, pp. 1–8.

[10] ROS. (2018). ROS industrial support for the KUKA LBR iiwa. [Online]. Available: http://wiki.ros.org/

[11] GitHub. (2018). ROS Indigo/Kinetic metapackage for the KUKA LBR IIWA R800/R820. [Online]. Available: http://github.com/IFL-CAMP/iiwa_stack/

[12] M. Safeea, R. Bearee, and P. Neto, "End-effector precise hand-guiding for collaborative robots," in *Advances in Intelligent Systems and Computing 694*, A. Ollero, A. Sanfeliu, L. Montano, N. Lau, and C. Cardeira, Eds. New York: Springer, 2018, pp. 595–605. doi: 10.1007/978-3-319-70836-2_49.

[13] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986. doi: 10.1177/027836498600500106.

Mohammad Safeea, Department of Mechanical Engineering, University of Coimbra, Portugal, and Arts et Métiers, ParisTech, France. E-mail: ms@uc.pt.

Pedro Neto, Department of Mechanical Engineering, University of Coimbra, Portugal. E-mail: pedro.neto@dem.uc.pt.