IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 21, 2022, accepted February 7, 2022, date of publication February 11, 2022, date of current version March 9, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3151070

Identifying Dynamic Parameters With a Novel Software Design for the M-DOF Collaborative Robot

ÖMER AYVACI[1], PAWEŁ SZULCZYŃSKI[2], AND MARCIN KIEŁCZEWSKI[2]
[1]Faculty of Computing and Telecommunication, Poznań University of Technology, 60965 Poznań, Poland
[2]Faculty of Control, Robotics and Electrical Engineering, Poznań University of Technology, 60965 Poznań, Poland

Corresponding author: Ömer Ayvaci (omer.c.ayvaci@doctorate.put.poznan.pl)

This work was supported by the Poznan University of Technology under the grant No. 0211/SBAD/0121.

ABSTRACT The primary goal of this project was to develop a general identification method via software that can be applied to collaborative robots. To achieve this, the collaborative ultralight robots Kinova Gen2 and Kuka LWR4+ with seven degrees of freedom (M-DOF) were used. Specifically, the ''recursive Newton-Euler'' formulation was used to provide a set of parameters that could describe the body structure and to create a general symbolic representation for collaborative robots. For parameter estimation, the least squares method was used. In addition, trajectories generated with random numbers typically do not produce consistent results; thus, verified trajectories were used. To verify trajectories, real robots were simulated with V-Rep before being executed. When untested trajectories are first tested on robots, undesirable results may occur. This method was convenient for parameter estimation and robot health; saves time; and increases the consistency of results. Algorithms were coded in MATLAB and ROS packages via Python. MATLAB, ROS, and V-Rep worked together in the Ubuntu operating system. The identification methods were modeled, implemented, tested, and validated successfully, and the results for both robots are reported in this article.

INDEX TERMS Collaborative robot, dynamic modeling, identification model, rigid body.

I. INTRODUCTION

A typical collaborative robot is a highly nonlinear multi-degree-of-freedom (M-DOF) rigid body mechanical system. Symbolic representations are included in the dynamic model calculated within the common configuration space. The model also includes multiple control inputs that are generated by the actuators included in the dynamic model. This type of model can be used to plan and control movement efficiently. Model-based control approaches, in particular, make it possible to improve the performance of the collaborative robot in terms of precision while also increasing the task execution speed [1].

The design and implementation of control algorithms necessitate an understanding of robot dynamics. It is common for collaborative robots to have dynamic models that are not precisely known beforehand. When computer-aided design (CAD) software is used to approximate some parameters, collecting real data during robot motion and using experimental identification techniques allow for further improvement of the robot model and modifying the robot model to be slightly based on CAD estimate circumstances. [2].

According to the relevant literature, identifying dynamic parameters depends on a few key factors. [3] Initially, the dynamic model of the manipulator is only a symbolic representation of the manipulator and is calculated based on fundamental parameters, which are also referred to as the miniature set of dynamic parameters that can be determined. Also, by ignoring the contribution of specific factors to joint torques or by taking advantage of specific topological features of the manipulator's construction, the dynamic model can be made smaller and more manageable. To obtain the best possible data collection for the estimated method in a sense that has not yet been defined, the manipulator trajectory must be set to an optimal value. It is thus necessary to compute the recommended values of the dynamic parameters using an appropriate estimation methodology.

This paper describes the various steps of the proposed identification procedure in greater detail. The identification

The associate editor coordinating the review of this manuscript and approving it for publication was Zheng H. Zhu.

software was developed via MATLAB and ROS. First, the symbolic representation is implemented in MATLAB, and then, the ROS collects the data from the real robot and is sent to MATLAB. Second, the data were processed in MATLAB and sent to V-Rep by ROS. This identification software runs on the Linux/Ubuntu operating system (now CoppeliaSim) [3]–[5].

The software enables the user to make changes to the environment to accommodate the robot. For example, the user may wish to identify all links or only certain links using DH parameters [6]. Additional extensions (e.g., a grinder, a hand) may be implemented by the user. For safety, the trajectory is first tested in the V-Rep simulation before being tested with a real robot because V-Rep allows the user to execute the robot in the simulation without harming the real robot and lets the user see the trajectory if sufficiently excited.

Finally, the procedure is tested experimentally to verify the simulation results.

A. DESCRIPTION OF ALGORITHM

The identification software has three different algorithm loops and one analysis step.

- The first loop performs symbolic calculations, where DH and vector parameters of the robot should be added to the *model.m* script by the user and run. The *model.m* script runs *kinematic.m* and *dynamic.m*, which represent the kinematics and dynamics of the robot. MATLAB derives the model of the robot kinematics and dynamics in the symbolic form, and saves the answer as a *regressor.mat*. Then, the software gives a notification that *regressor.mat* has been successfully created and asks the user if it allows running the *fullregressor.m* script. If no, the loop repeats. If yes, *fulllregressor.m* loads *regressor.mat* and uses an optimization algorithm to optimize the model of the robot dynamics and saves the answer as a *fullregressor.mat*; then, the first loop ends successfully.
- The second loop plans the trajectory, which executes the simulation and the real robot regarding the planned trajectory. The user should create a trajectory in the *trajectory.m* script or simulate a trajectory via V-REP and save the trajectory into the *trajectory.m* file. V-Rep provides the user with a chance to see whether the final trajectory is harmful to the robot's structure. If the simulated trajectory appears harmful or does not satisfy the user, this step must be repeated. Then, the *trajectory.m* script is run by the user, and MATLAB sends data to the ROS and runs the real robot. Real data are collected by the ROS and saved in real time as a *.txt* file (e.g., *position.txt*, *velocity.txt*). When the execution finishes, and real data are saved, the software asks the user to save the *.txt* file as a *.mat* file. If no, the loop ends, which means that the *.mat* file is not created, but this part can be done manually later on. If yes, *trajectory.mat* is created, which means that the second loop ends successfully.
- The third loop defines the dynamic parameters. The *parameters.m* script identifies relevant parameters, and the user runs the script, which loads *fullregressor.mat* from the first loop and *trajectory.mat* from the second loop. The kinematically and dynamically created symbolic robot executes the trajectory and saves the answers in two different *.mat* files. First, *fullnumericregressor.mat* stores information about the dynamic parameters, which are not yet categorized or estimated as identifiable, non-identifiable, and linear combinations; this estimation and categorization will be done slightly later. Second, *estimatedmotion.mat* stores information about the estimated position, velocity, acceleration, and torque. Then, the software prints a notification that *fullnumericregressor.mat* has been successfully created and asks the user if to run *identification.m* script. If no, the loop repeats. If yes, the *identification.m* script loads *fullnumericaregressor.mat*, estimates the parameters using the Least Square method, and saves the results as *parameters.mat*. This *.mat* file contains information about parameters including X_i (identifiable parameters), X_{ni} (non-identifiable parameters), and X_{lc} (linear combinations), and reports them as output, ending the third loop.

Finally, we analyze the collected and symbolically generated data. This last part is called the validation model, which runs the *myplot.m* script. This script loads the *trajectory.mat* and *estimatedmotion.mat* runs and compares the estimated and measured motions in addition to reporting information about the identification of dynamic parameters. The accuracy of the comparison between the estimated and measured motions shows that the identified parameters are correct and that the validated model performs as expected.

II. MATERIALS AND METHODS

- For the first experiment, a Kinova Gen2 7-DOF lightweight robot was used [7], [8]. This robot has a maximum payload of 2.4 kg and a total weight of 5.5 kg.
- For the second experiment, a Kuka LWR4+ lightweight robot was used [9]. This robot is a 7-DOF lightweight robot that is similar to the Kinova Gen2, and has a payload of 7 kg and a total weight of 16 kg.
- Tools and software packages: ROS Melodic Morenia, Ubuntu 18.04 (Bionic), VREP Release 3.5.0, MATLAB 2015b, Python 3.8.
- Computer features: Intel Core i7 4710MQ CPU @2.50 GHz, Intel Haswell HM87 Chipset, DDR3 2×8GB(800 MHz) Ram, Intel HD Graphics 4600 GPU, Kingston mS200 120GB M.2 SSD @520MB/s write, 550MB/s read.
- For the identification method, this software was created from four different perspectives. First, the symbolic representation was programmed in MATLAB. Second, V-Rep was used for trajectory simulation and validation. Then, the validated trajectory was used to

execute the trajectory in the real robots. Third, parameter estimation was used to identify dynamic parameters. Fourth, we validated the model by comparing the estimated and measured data.

A. MODEL DERIVATION

We now review the conditions of movement for an open-chain controller made out of n inflexible connections, which are obtained using the recursive Newton-Euler formalism [10]. We consider that the Denavit-Hartenberg (DH) [11], [12] notation is used to identify the frame i^{th} associated with the relevant link and that the rotation matrix describing the orientation of frame k with respect to frame j is denoted by R_{jk}. With rotating joints, the equations of motion for the frame I connected to the link, with its origin on the axis of the joint $i+1$, can be composed as follows [13].

We consider the case of an open-chain manipulator composed of rigid links connected by joints and eventually by rigid gears. Using the recursive Newton-Euler formulation [10], it is feasible to determine the equations of motion for a system with a given set of initial conditions. Using the Denavit-Hartenberg notation [14], you could choose, for example, the frame I that is linked to the relevant link and then express the rotation matrix that indicates the orientation of frame k concerning frame j in R_{jk}. With rotating joints, it is necessary to write out the equations of motion for the frame connected to the first link, which has its origin on the axis of joint $i+1$, in the following form [13]:

$$\omega_i^i = \left(R_i^{i-1}\right)^T \left(\omega_{i-1}^{i-1} + \dot{q}_i z_0\right), \tag{1}$$

$$\dot{\omega}_i^i = \left(R_i^{i-1}\right)^T \left(\dot{\omega}_{i-1}^{i-1} + \dot{q}_i \omega_{i-1}^{i-1} \times z_0 + \ddot{q}_i z_0\right), \tag{2}$$

$$\dot{v}_i^i = (R_i^{i-1})^T \left(\dot{v}_{i-1}^{i-1} + \dot{\omega}_{i-1}^{i-1} \times r_{i-1,i}^i + \right. \\ \left. + \omega_i^i \times \left(\omega_i^i \times r_{i-1,i}^i\right)\right), \tag{3}$$

A vector is represented in a certain frame, denoted by the superscript. If v_i is the linear velocity of the frame origin i, then ω_i is the angular frame velocity, and q_i is the joint variable in the preceding equations i,and is the joint variable. Then, the unique terms for edge i are fulfilled:

$$f_i^i = R_{i+1}^i f_{i+1}^{i+1} + m_i \dot{v}_i^i + \dot{\omega}_i^i \times m_i r_{i,ci}^i \\ + \omega_i^i \times \left(\omega_i^i \times m_i r_{i,ci}^i\right), \tag{4}$$

$$\mu_i^i = R_{i+1}^i \mu_{i+1}^{i+1} + r_{i-1,i}^i \times f_i^i + m_i r_{i,ci}^i \times \dot{v}_i^i \\ + I_i \dot{\omega}_i^i + \omega_i^i \times I_i \omega_i^i, \tag{5}$$

$$\tau_i = \left(R_{i-1}^i \mu_i^i\right)^T z_0 + I_{mot,i} \ddot{q}_i + \tau_{f,i}, \tag{6}$$

where $z_0 = [0\ 0\ 1]^T$ is a unit vector, force applied is the f_i at the frame origin i, momentum applied is the μ_i connected to link, and torque applied is the τ_i connected to link. The link mass is m_i; the vector position is $r_{i-1,i}$ which is from the center of the body; the center point of mass is i connected to link i; the vector position $r_{i-1,}$ is from the center of frame $i-1$ to the frame origin i; and the inertia tensor of the matrix expression is I_i with regard to edge i. With the friction torque $\tau_{f,i}$ and $I_{mot,i}$ being the inertia of the i^{th} actuator at the joint hub, a few mathematical equations are suggested in [15] and [16]; this model is used here for identification purposes:

$$\tau_{f,i} = b_{s,i} \operatorname{sign}(\dot{q}_i) + b_{v,i} \dot{q}_i, \ |\dot{q}_i| < \dot{q}_{s,i}. \tag{7}$$

The expression $\dot{q}_{s,i}$ represents a sufficient velocity limit; the expression $b_{s,i}$ represents static friction, and the expression $b_{v,i}$ represents viscous friction. Due to the difficulty inherent to deriving a simple model for static friction, the friction is considered to be zero if $|\dot{q}_i| < \dot{q}_{s,i}$; this friction model is linear in $b_{s,i}$ and $b_{v,i}$. . As shown by [1], [17]–[19] and [20], the equations of motion are linear in terms of the dynamic parameters, and thus, we obtain:

$$\tau = P(q, \dot{q}, \ddot{q})\, \gamma. \tag{8}$$

The top triangular matrix is $P(q, \dot{q}, \ddot{q})$, whereas $(n \times 11n)$ is only determined by kinematic variables and $\gamma \in R^n$ is the dynamic parameter vector for each link, which includes the following:

- Mass (1 unknown),
- First-order moment (3 unknowns),
- Inertia matrix (6 unknowns),
- Actuator inertia (1 unknown).

The identification software first calculates the origin of the frames and the center of mass with the scale multiplied with respect to DH parameters. Setting frames for each link in the Newton-Euler definition is a smart method because it frees all length vectors during robot design.

The irregular shapes of the links in collaborative robots make it difficult to estimate some identification parameters. Thus, parameter estimation was the most challenging aspect of this study. To keep things simple, we modelled the links as cylinders with a uniform mass density, and each link is a cylinder in its geometric center, where the center of mass is located.

$$\begin{aligned} R_{01} &= \begin{bmatrix} 0 & R_{01y} & 0 \end{bmatrix} \\ R_{12} &= \begin{bmatrix} 0 & R_{02y} & 0 \end{bmatrix} \\ R_{23} &= \begin{bmatrix} R_{23x} & 0 & 0 \end{bmatrix} \\ R_{34} &= \begin{bmatrix} 0 & R_{34y} & 0 \end{bmatrix} \\ R_{45} &= \begin{bmatrix} 0 & R_{45y} & R_{45z} \end{bmatrix} \\ R_{56} &= \begin{bmatrix} 0 & 0 & R_{56z} \end{bmatrix} \\ R_{67} &= \begin{bmatrix} 0 & 0 & R_{67z} \end{bmatrix} \end{aligned} \tag{9}$$

All pivot frameworks can be determined by the results of essential turns about the z-hub and the x-hub, where these fundamental revolution matrices are characterized as a rule structure. In general, rotation matrices may be computed as combinations of simple rotations around the z-axis and the x-axis, with the simplest rotation matrices being defined as

follows [21]:

$$R_{z,\theta} = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (10)$$

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix}. \quad (11)$$

The following multiplication results using a recursive procedure can be derived:

$$R_j^i = R_{j-1}^0 R_j^{j-1} \quad j = 2, \ldots, 7 \quad (12)$$

B. OPTIMIZATION

γ, which comes from the equation of motion (8) which may be separated into three classes [22]:

- Identifiable
- Non-identifiable
- Identifiable as a linear combination

The identifiable parameters are P linear independent columns and are only in linear combination with the other parameters, where the second class of parameters may make any contribution to the joint torques. This result indicates that a linear combination of the columns of P can be used to replace them. It is possible to remove non-identifiable parameters from the model because they do not contribute to the torque exerted at the joints.

Thus, the γ vector can be supplanted by a base parameter vector $\theta \in R^p$ with $p < 11n$, and the matrix P can be supplanted by comparing $(n \times p)$, which is decreased matrix W [23], [24]:

$$\tau = W(q, \dot{q}, \ddot{q})\theta. \quad (13)$$

The representative of W (13) is found based on the robot's kinematics. Before estimating the identifiable parameters of the real robot, the deletion of γ minimizes the points of the parameters but also minimizes the error in the robot's computation model.

C. TRAJECTORY AND V-REP PLANNING

1) EXCITING TRAJECTORY

The least-squares calculation uses the excitation matrix $II_N = W_N^T W_N$, where the relapse matrix W_N is used to minimize the condition number and maximize the singular value.

The exciting trajectory planning is characterized as many estimation focuses $\rho(t_i)\,(i = 1 \ldots N)$, which are points that might be acquired by limiting the cost function [23], [25]:

$$f(\xi(t_i)) = \lambda_1 \text{cond}(W_N) + \lambda_2 \frac{1}{\sigma_0(W_N)}, \quad (14)$$

where the scalar λ_1 and λ_2 address the general loads between the condition number of the perception matrix $cond\,(W_N)$ and is a small singular value: $\sigma_0(W_N)$ [17].

The square of each single estimate of (W_N) is in monotonous expansion with the number of rows [26]. The points of estimation by taking all the examples would upgrade by gathering along the interpolated trajectory. Indeed, the N unique examples in $t_I \ldots t_N$. However, another W_* (e.g., the regression matrix) can evaluate the layout of the smallest square in more detail:

$$\theta = \left(W_*^T W_*\right)^{-1} W_*^T \tau_*, \quad (15)$$

where W_* is a matrix underlying a similar path as W_N, but uses all of the estimations gathered along the added direction, which also ensures that the solitary subsequent term diminishes, while nothing can be closed on the conduct of the initial term when the number of tests is expanded. Similarly, the extra examples to be added to the interjected direction can be chosen by the following rules:

$$\text{if f}(\zeta(t_1), \ldots, \zeta(t_K)) - f(\zeta(t_1), \ldots, \zeta(t_{K-1})) > \eta, \quad (16)$$

where η is a reasonably chosen edge. Then, all the parts $W(\zeta(t_i))$ in the matrix W_* relating to the disposed of information are wiped out. Importantly, the choice of tests as indicated by (15) can create associating and distorted experiences due to undersampled data, evading operation out a prior low-pass digital filtering of the grouping of the progression of data. $\zeta(t_i)$

The contribution of the proposed algorithm is the representative articulation of the matrix W for the joint positions, speeds, and increasing velocities as far as possible. The yield of the ideal arrangement of estimation focuses on $\delta(t_1) \ldots . \delta(t_N)$.

The interpolation algorithm is actualized to provide a smooth direction. In the given approach, fifth-order polynomials are used for each positional segment of the N estimation. Every one of the $N-1$ time intervals corresponds to the polynomial coefficients, and every joint is determined by the forcing coherence of $(q, \dot{q}, \ddot{q})$. In each $N-1$ time interval $(t_j, t_{j+1})\,(j = 1, \ldots, N-1)$. The direction for the I_{th} joint in the j_{th} time intervals can thus be described as:

$$\xi = \begin{bmatrix} q(i) \\ \dot{q}(i) \\ \ddot{q}(i) \end{bmatrix} = \begin{bmatrix} t^5 & t^4 & t^3 & t^2 & t & 1 \\ 5t^4 & 4t^3 & 3t^2 & 2t & 1 & 0 \\ 20t^3 & 12t^2 & 6t & 2 & 0 & 0 \end{bmatrix} \times p. \quad (17)$$

However, the maximum values of $(q, \dot{q}, \ddot{q})$ in each range may exceed the mechanical limits of the joints. In this situation, the few interval amplitudes $\Delta T_j = t_{j+1} - t_j\,(j = 1, \ldots N-1)$ may be changed to find an iterative set of polynomials that meet the common limits for positions as much as possible for position, velocity and acceleration.

The single polynomial is used because the MATLAB plots must be in the continuous form from an initial time to end

time. A matrix of this type is shown below:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 5 & 4 & 3 & 2 & 1 & 0 \\ 20 & 12 & 6 & 2 & 0 & 0 \end{bmatrix}. \tag{18}$$

- The first three rows describe the ''initial time''s of position, velocity, and acceleration.
- The second three rows describe the ''final time''s of position, velocity, and acceleration.

Scaling the time and limiting the velocity and position, the safety coefficient was used. In this case, a safety coefficient ($\propto$) is set to 0.2 because 5^{th}-order polynomials are used. Safety coefficient ($\propto$) scales the velocity to 0.2. Using V-Rep planning, we see that the estimated velocity is harmful to the robot.

After applying the safety coefficient ($\propto$), the system of equations for each step (position, velocity, and acceleration) became the following system of equations:

$$\begin{aligned} &\xi(\propto t) \\ &= \begin{bmatrix} q\,(i) \\ \dot{q}\,(i)\,(\propto) \\ \ddot{q}\,(i)\,(\propto^2) \end{bmatrix} \\ &= \begin{bmatrix} \propto^5 t^5 & \propto^4 t^4 & \propto^3 t^3 & \propto^2 t^2 & \propto t & 1 \\ \propto^4 5t^4 & \propto^3 4t^3 & \propto^5 3t^2 & \propto 2t & 1 & 0 \\ \propto^3 20t^3 & \propto^2 12t^2 & \propto 6t & 2 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{19}$$

This case was for the Kinova Gen2 or Kuka LWR4+ robot. Other robots may not require a safety coefficient. Thus, the user may neglect the safety coefficient function, or if the user still needs the safety coefficient, the user can change its value.

2) V-REP PLANNING

In this case, remote API functions in MATLAB were used. The API helps the cooperation of MATLAB and V-Rep in ROS under the Linux/Ubuntu operating system [27], [28]. Importantly, the framework is checked by V-Rep due to security issues prior to using a genuine controller. [29] V-Rep is an open-source reenactment that gives a practical near genuine outcome. Similar powerful models of the Kinova Gen2 and Kuka LWR4+ robots were executed in the V-Rep simulation. However, an exciting trajectory can also be created via V-Rep. Using the Gazebo [30] package, which works under ROS, the system can submit the exciting trajectory into the MATLAB script [28], [31].

Gen2 and LWR4+ exist in the V-Rep library, which has been used for the simulation. To explain the robot simulation in V-Rep [32], we must examine the structure of the robot.

The joint connects two body parts; the body part closest to the trunk is considered fixed, and the body part farther away from the trunk is considered to rotate around the joint axis. Each joint has a frame attached to it, which allows the body

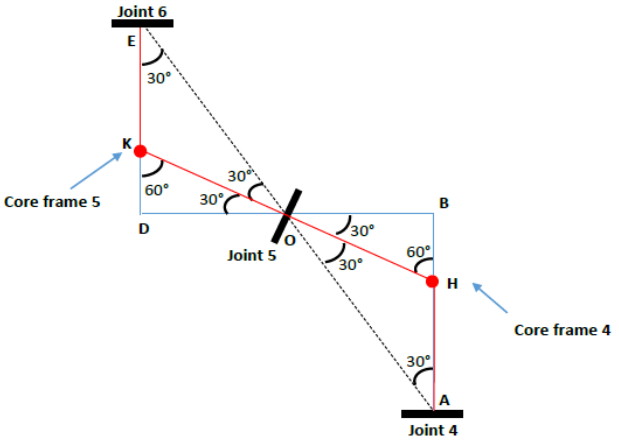

FIGURE 1. Joints of core frames.

TABLE 1. DH → V-REP.

DH	V-REP
θ_{1V-REP}	$-\theta_{1dh}$
θ_{2V-REP}	$\theta_{2dh} + 180°$
θ_{3V-REP}	θ_{3dh}
θ_{4V-REP}	θ_{4dh}
θ_{5V-REP}	θ_{5dh}
θ_{6V-REP}	$\theta_{6dh} - 180°$
θ_{7V-REP}	θ_{7dh}

TABLE 2. V-REP → DH.

V-REP	DH
θ_{1dh}	$-\theta_{1V-REP}$
θ_{2dh}	$\theta_{2V-REP} - 180°$
θ_{3dh}	θ_{3V-REP}
θ_{4dh}	θ_{4V-REP}
θ_{5dh}	θ_{5V-REP}
θ_{6dh}	$\theta_{6V-REP} + 180°$
θ_{7dh}	θ_{7V-REP}

parts to be rotated. At the initial position, all joint frames are oriented in the same direction as the robot's body. It is then necessary to perform roll and pitch rotations around the X and Y coordinate systems, as well as yaw rotation around the Z coordinate system.

Considering the triangle AOB, we obtain:

$$OB = \sin\left(30°\right) \times OA, \tag{20}$$

where $OA = d_4$.

Now, considering the triangle OBH, we have:

$$OH = \frac{OB}{sin(60°)} = \frac{\sin(30°)}{d_4} sin(60°) = AH = OK = KE. \tag{21}$$

The current configuration figure (2) on the left is described in Tables (1), and (2). This current configuration exists in the V-Rep library.

We are required to convert the current configuration to the chosen start configuration to apply our exciting trajectory planning shown in the following figure (2) on the right, which

TABLE 3. DH → V-REP.

DH	V-REP
θ_{1V-REP}	$+90°$
θ_{2V-REP}	$+150°$
θ_{3V-REP}	$0°$
θ_{4V-REP}	$+270°$
θ_{5V-REP}	$+70°$
θ_{6V-REP}	$+90°$
θ_{7V-REP}	$+90°$

TABLE 4. V-REP → DH.

V-REP	DH
θ_{1dh}	$-90°$
θ_{2dh}	$-30°$
θ_{3dh}	$0°$
θ_{4dh}	$+270°$
θ_{5dh}	$+70°$
θ_{6dh}	$+270°$
θ_{7dh}	$+90°$

is described in the following table (3), (4). These adjustments were made to provide a more exciting trajectory, which may not be needed by the user; th e user's qualifications should determine the exciting trajectory. However, if the user's robot is not yet included in V-Rep, the user may easily add it. (For more information, see [28], [32])

3) COLLECTING DATA

Last, the exciting trajectories are executed on a real robot using MATLAB, while ROS records in real time at a 100-Hz sampling rate. To analyze the collected data, a low-pass digital filter was used:

$$\dot{X}_k = \frac{x_{k+1} - x_{k-1}}{\Delta T}. \tag{22}$$

D. PARAMETER ESTIMATION

We consider N time instants $t_1, \ldots, t_N$, where $nN > p$. By stacking the matrix together, the perception matrix can be shaped as follows [33]

$$\tau_N = \begin{bmatrix} \tau\,(t_1) \\ \tau\,(t_2) \\ \vdots \\ \tau\,(t_N) \end{bmatrix} = \begin{bmatrix} W(\xi\,(t_1)) \\ W(\xi\,(t_2)) \\ \vdots \\ W(\xi\,(t_N) \end{bmatrix}, \tag{23}$$

where the vector $\xi\,(t_i) \in R^{3n}$ characterizes the ''measurement point'' at time t_i, W_N signifies the full-position matrix $(Nn \times p)$ obtained from W figured in the N estimated focuses and $\tau_N \in R^{Nn}$ is the vector acquired from the N estimated force $\tau\,(t_i)$.

Hypothetically, as long as the determinant of the perceptual framework W_N, which relies upon the exciting trajectory that has been used in the identical parameters, is non-zero, the obscure parameters θ can be assessed by the notable least-squares/weighted least-squares estimation equation:

$$\theta = \left(W_N^T W_N\right)^{-1} W_N^T \tau_N. \tag{24}$$

TABLE 5. Representation of $(n \times 11n)$.

First	Seventh
$\varphi_1 = m_1$	$\varphi_{67} = m_7$
$\varphi_2 = mc_{1,x}$	$\varphi_{68} = mc_{7,x}$
$\varphi_3 = mc_{1,y}$	$\varphi_{69} = mc_{7,y}$
$\varphi_4 = mc_{1,z}$	$\varphi_{70} = mc_{7,z}$
$\varphi_5 = I_{1,xx}$	$\varphi_{71} = I_{7,xx}$
$\varphi_6 = I_{1,xy}$	$\varphi_{72} = I_{7,xy}$
$\varphi_7 = I_{1,xz}$	$\varphi_{73} = I_{7,xz}$
$\varphi_8 = I_{1,yy}$	$\varphi_{74} = I_{7,yy}$
$\varphi_9 = I_{1,yz}$	$\varphi_{75} = I_{7,yz}$
$\varphi_{10} = I_{1,zz}$	$\varphi_{76} = I_{7,zz}$
$\varphi_{11} = I_{1,m}$	$\varphi_{77} = I_{7,m}$

Because the measured torque is affected by the measurement noise, a limitation must be placed on the validation path to ensure an identical result. Criteria have been proposed in the following publications (refer to [1]). In this study, the limitation of the number of conditions and the expansion of the minimum singular value of W_N as in [20] is implemented.

Considering that if there is a possibility that any prior data of dynamic parameters are available, a weighted pseudoinverse of W_N may be used to improve the estimation for each parameter. For this reason, a weighted pseudoinverse of W_N was used [17].

Dynamic parameters are avoidable from the prior data around the assessments. Generally, an unclean inverse of W_N can be used [17] to improve the adequacy of the appraisal on each parameter.

The least-squares recursive calculation may be used rather than the clustering equation (23), and the plan of assessment should be updated by taking all the models accumulated along the additional course, despite the N novel models in $t_1, \ldots, t_N$. Then, another regressor cross section W_* can be attempted to survey the least-square plan.

The models in table (5) are shown for the second → seventh joint measurements. Each joint has eleven parameters where $(n \times 11n)$. The programming considers the outcomes under three cases that are ''identifiable alone(X_i), non-identifiable(X_{ni}) and as linear combinations(X_1 X_2)''. Capacities, which are ''X_i, X_{ni}, X_1, X_2, A ''are expected to identify the disposition of the parameters required to perform the decreased regression. Therefore, in the SVD-based product [34] assurance of up-to-date basic parameters, see [35], [36]. Thus, the software uses the irregular direction for exciting all the parameters and registering the full regressor, which assembles the vectors with the records of the parameters identifiable alone or non-identifiable. Then, the framework deletes the related segments from the regressor to distinguish between straight compounds. Eventually, the matrix checks if there is any consistency fizzled. The results are then saved as a *.mat* file by the designed software.

The models in table (5) are shown for the second → seventh joint measurements. Each joint has eleven parameters where $(n \times 11n)$. The programming considers the outcomes under three cases that are ''identifiable alone(X_i),

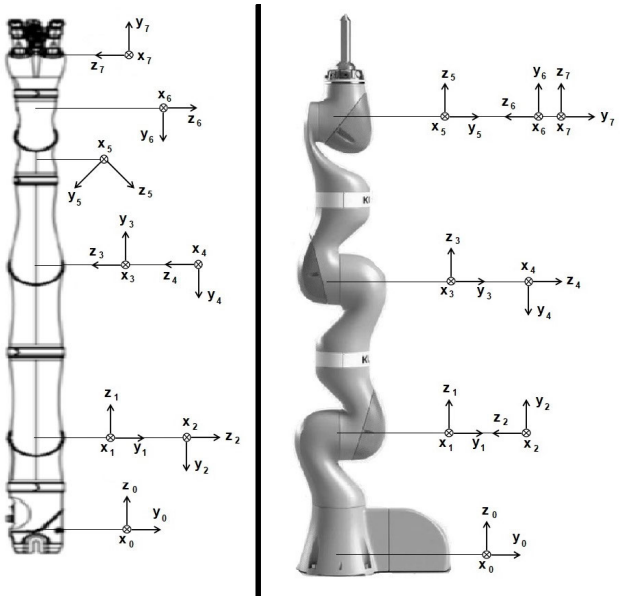

FIGURE 2. Symbolic representation of Kinova Gen2 (left) and Kuka LWR4+ (right).

non-identifiable(X_{ni}) and as linear combinations(X_1 X_2)''. Capacities, which are ''X_i, X_{ni}, X_1, X_2, A ''are expected to identify the disposition of the parameters required to perform the decreased regression. Therefore, in the SVD-based product [34] assurance of up-to-date basic parameters, see [35], [36]. Thus, the software uses the irregular direction to excite all the parameters and to register the full regressor, which assembles the vectors with the records of the parameters identifiable alone or non-identifiable. Then, the framework deletes the related segments from the regressor to distinguish between straight compounds. Eventually, the matrix checks if there is any violation of consistency. The results are then saved as a *.mat* file by the designed software.

E. VALIDATION MODEL

The validated model was analyzed, and identifiable parameters, CAD, and validated trajectory are discussed in the results section.

III. RESULTS

Symbolic representations of the ultralight collaborative robots Kinova Gen2 and Kuka LWR+4 are shown in figure (2). DH parameters were added to the MATLAB script and are shown below for both robots in Tables (6) and (7)

The identifiability of the dynamic parameters of Gen2 and LWR4+ is shown in table (8), and (9) where a green block indicates identifiable alone (X_i) parameters, a red block indicates non-identifiable (X_{ni}) parameters, and an empty block indicates linear combinations (X_{cl}). For instance, for Gen2, With only 77 parameters (11 parameters for each joint), we find ''identifiable alone'' (X_i) = 41, ''non-identifiable''(X_{ni}) = 13 and ''linear combinations'' (X_1) = 14, (X_2) = 9, which means a total of 77 parameters (consistent)

TABLE 6. DH parameters for Kinova Gen2.

$a(m)$	$\alpha(deg)$	$d(m)$	θ
0	$+90°$	0.2755	θ_1
0	$+90°$	0	θ_2
0	$+90°$	-0.41	θ_3
0	$+90°$	-0.0098	θ_4
0	$+90°$	-0.3111	θ_5
0	$+90°$	0	θ_6
0	$0°$	0.2638	θ_7

TABLE 7. DH parameters for Kuka LWR+4.

$a(m)$	$\alpha(deg)$	$d(m)$	θ
0	$+90°$	0	θ_1
0	$-90°$	0	θ_2
0	$-90°$	0	θ_3
0	$+9°$	0	θ_4
0	$+90°$	0	θ_5
0	$-90°$	0	θ_6
0	$0°$	0	θ_7

TABLE 8. Identifiability of the parameters of Gen2.

Gen2	Number of Joints						
Base Parameters	1	2	3	4	5	6	7
m							
mcx							
mcy							
mcz							
Ixx							
Ixy							
Ixz							
Iyy							
Iyz							
Izz							
Im							

TABLE 9. Identifiability of the parameters of LWR4+.

LWR4+	Number of Joints						
Base Parameters	1	2	3	4	5	6	7
m							
mcx							
mcy							
mcz							
Ixx							
Ixy							
Ixz							
Iyy							
Iyz							
Izz							
Im							

when ($n \times 11n$). For example, if we consider the first joint, there are no identifiable parameters that are fixed to the ground due to yaw rotation. There is only one parameter, which is a ''linear combination'' (φ (11) = I1m). All parameters were examined in detail, and as shown in the contrasts between assessment and experienced outcomes, mistakes were limited by using condition [36].

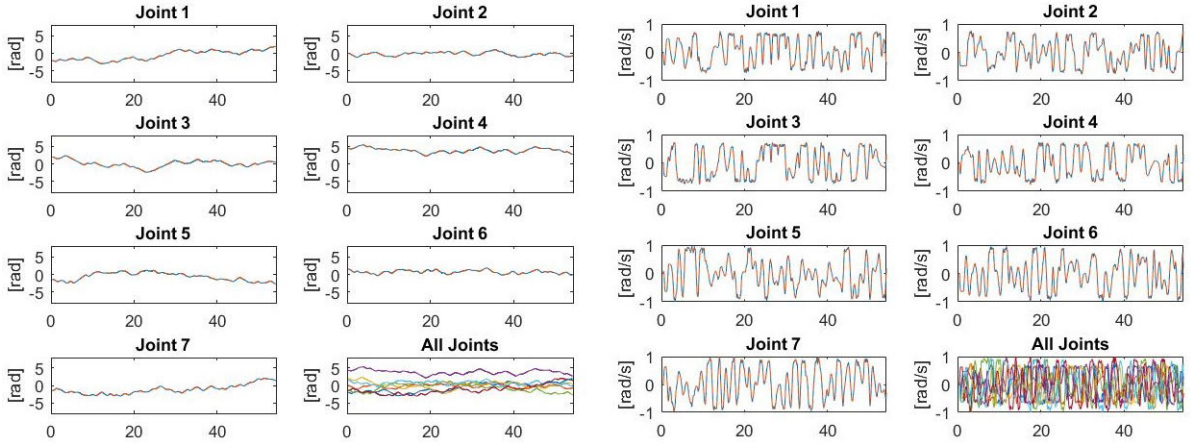

FIGURE 3. Estimated and measured position (left),and velocity (right) of Gen2.

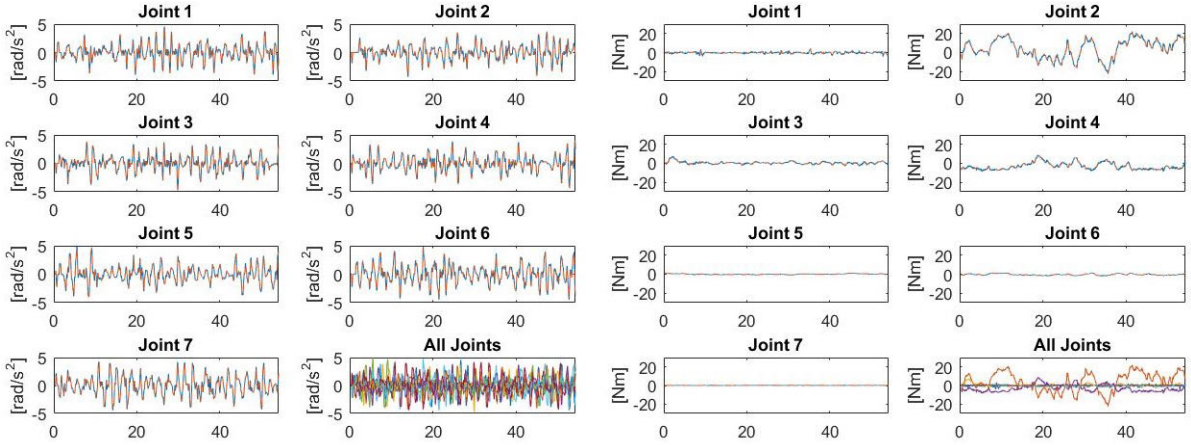

FIGURE 4. Estimated and measured acceleration(left),and torque (right) of Gen2.

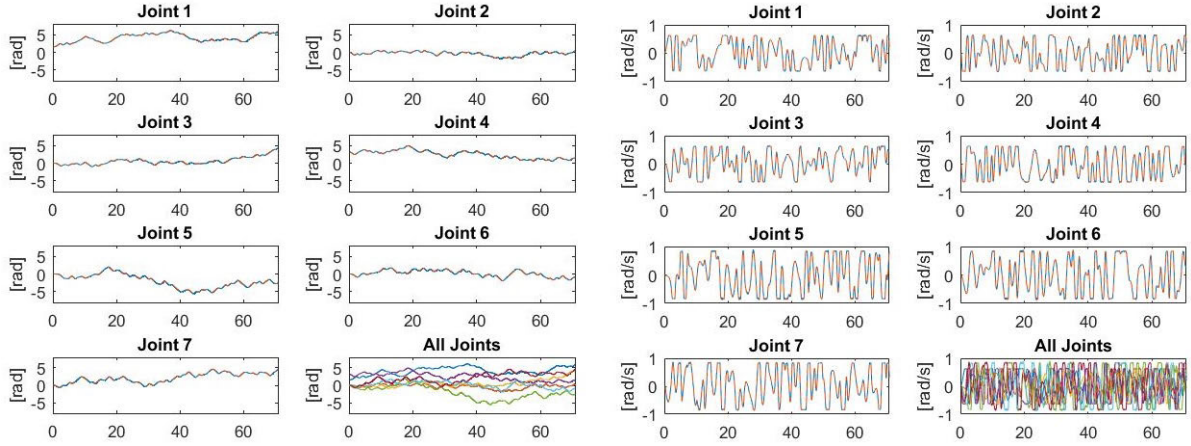

FIGURE 5. Estimated and measured position(left),and velocity (right) of LWR4+.

SLAMBench scores for different setups are shown in table (11) [37], [38].

- **LU:** *Perform "LU matrix factorization" of a full matrix*,
- **FFT:** *Perform "Fast Fourier transform" of a full vector*,
- **ODE:** *Solve van der Pol equation with "Solve non-stiff differential equations medium order method"*,
- **Sparse:** *Solve a symmetric sparse linear system*,
- **2D:** *Plot Lissajous curves*,

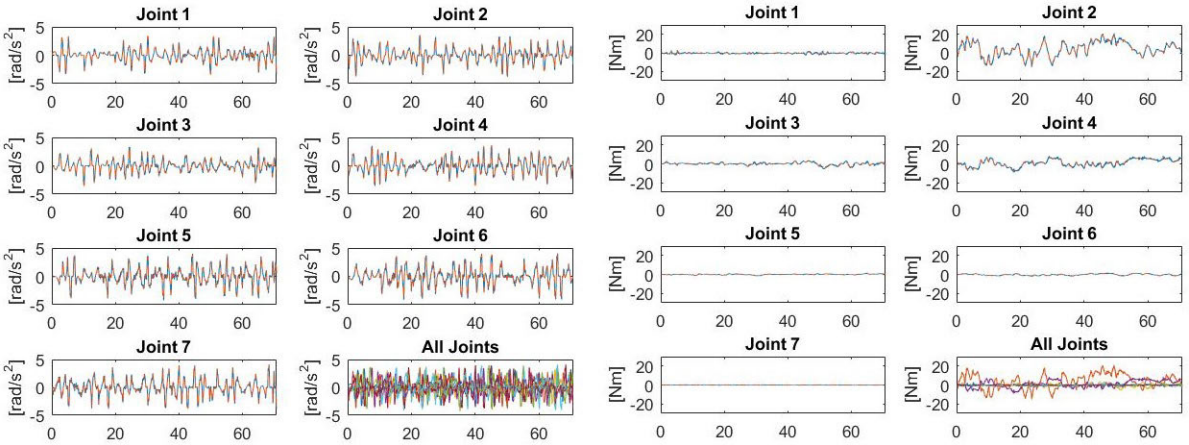

FIGURE 6. Estimated and measured acceleration (left),and torque (right) of LWR4+.

TABLE 10. Values of base parameters for Gen2 and LWR4+.

Values for dynamic parameters on CAD and validated trajectory							
Kinova Gen2				*Kuka LWR4+*			
Parameters	CAD	Value	σ	Parameters	CAD	Value	σ
φ_8	-0.005507	-0.397151	2.10e-04	φ_8	0.100504	0.064544	2.09e-04
φ_{11}	0.010349	-0.274931	2.99e-04	φ_{11}	-0.120586	0.041032	7.94e-04
φ_{16}	-0.033926	-0.122219	8.56e-04	φ_{16}	-0.186128	0.122979	9.00e-04
φ_{19}	-0.001927	-0.028627	1.56e-04	φ_{19}	-0.106644	0.162274	2.41e-04
φ_{21}	-0.004843	-0.037547	4.30e-03	φ_{21}	0.020081	0.023511	8.66e-03
φ_{22}	-0.027155	-0.064756	5.72e-02	φ_{22}	0.099566	0.261741	8.00e-02
φ_{23}	0.015015	0.526268	1.8e02	φ_{27}	0.286927	0.286026	6.9e02
φ_{24}	0.006377	0.135123	1.1e03	φ_{30}	0.080716	0.186558	2.1e03
φ_{30}	-0.022705	-0.316972	3.8e-04	φ_{32}	0.206210	0.099467	1.2e-04
φ_{32}	0.029082	0.222503	9.3e-03	φ_{38}	0.050335	0.560661	2.2e-03
φ_{38}	0.001575	0.087332	9.3e-03	φ_{41}	0.030381	-0.374102	2.2e-03
φ_{41}	0.006102	0.149891	7.7e-03	φ_{43}	0.080716	-0.186558	1.1e-03
φ_{43}	-0.005988	-0.458723	4.22e-04	φ_{49}	0.192886	0.375134	1.1e-04
φ_{49}	-0.005258	-0.037407	4.2e-04	φ_{52}	0.170100	0.094557	1.8e-04
φ_{52}	-0.001022	-0.001976	2.9e-02	φ_{53}	0.013155	0.161991	4.5e-02
φ_{53}	-0.022713	-0.456223	2.9e-02	φ_{54}	0.022786	0.280577	4.9e-02
φ_{54}	-0.530109	-0.073514	6.6e-03	φ_{60}	-0.298553	0.057150	1.0e-03
φ_{60}	-0.104065	-0.003637	1.7e-03	φ_{63}	0.420432	0.057924	2.2e-03
φ_{63}	0.008940	0.569698	1.7e-03	φ_{64}	0.476370	-0.142085	2.4e-03
φ_{64}	-0.034285	-0.111745	2.0e-03	φ_{65}	0.070366	-0.234556	2.7e-03
φ_{65}	-0.098214	-0.551925	9.8e-02	φ_{71}	0.121878	0.406264	6.0e-02
φ_{71}	-0.016258	-0.892550	2.6e-04	φ_{74}	0.461059	-0.058182	1.7e-04
φ_{74}	-0.197550	-0.320582	2.7e-04				

TABLE 11. SLAMBench scores comparison for the software on different setups.

Computer Type	LU	FFT	ODE	Sparse	2D	3D
Windows 7(64-bit), Intel Xeon E5-265 v2 @2.6 GHz	0.1324	0.0672	0.1154	0.1378	0.2800	0.3062
Windows 7(64-bit), Intel Xeon E5-2665 0 @2.4 GHz	0.1271	0.0706	0.1295	0.1554	0.3415	0.4429
Surface Pro 3, Windows 8.1, Intel Core i5-4300U @1.9 GHz	0.2001	0.0854	0.0591	0.1303	0.7433	0.6018
Linux (64-bit), Intel Xeon E5-2665 0 @2.4 GHz	0.1355	0.0658	0.1382	0.1285	0.7113	0.6994
Mac mini, OSX 10.10.3 (64-bit) Intel Core i3 @2.3 GHz	0.1709	0.0749	0.1081	0.1211	0.7348	0.8031
This Machine	0.2340	0.1116	0.0608	0.1571	0.5196	0.7608
Macbook Pro, OSX 10.9.5 (64-bit) Intel Core i5 @2.6 GHz	0.1863	0.0900	0.0571	0.1234	1.8211	1.9371
Linux (64-bit), Intel Core2 Quad Q9300 @2.50 GHz	0.6504	0.2218	0.1684	0.2958	0.9702	0.8848

- **3D:** *Display color-mapped "Peaks function" with clipping and transforms.*

Figures (3), and (4) show the reconstruction errors for Gen2 between the measured and estimated positions, velocities, accelerations, and torques on an exciting trajectory for validation. Additionally, Figures (5), and (6) show the same comparison for LWR4+. CAD estimation, measured value and reconstruction errors (σ) on validated trajectory details are shown for each identified base parameter in table (10). Reconstruction errors were small and numerically similar,

TABLE 12. Loaded computation time for each loop.

Joints	Gen2	LWR4+
$Joint_1$	15.90 Nm	12.25 Nm
$Joint_2$	7.58 Nm	7.93 Nm
$Joint_3$	4.74 Nm	5.22 Nm
$Joint_4$	8.57 Nm	7.12 Nm
$Joint_5$	8.59 Nm	8.33 Nm
$Joint_6$	3.93 Nm	4.01 Nm
$Joint_7$	2.94 Nm	2.99 Nm

TABLE 13. Prediction error for each joint.

Loops	Gen2	LWR4+
Loop 1	165.06 sec	180.72 sec
Loop 2	183.2 sec	208.4 sec
Loop 3	109.2 sec	122.52 sec
Total Time	457.46 sec	511.64 sec

TABLE 14. Unloaded computation time for each loop.

Loops	Gen2	LWR4+
Loop 1	6.12 sec	6.79 sec
Loop 2	6.85 sec	7.3 sec
Loop 3	3.19 sec	3.28 sec
Total Time	16.16 sec	17.37 sec

which means that comparisons between estimated and measured values were consistent for both robots. This consistency indicates that the software operates well. In addition, the joint prediction error for Gen2 and LWR4+ is shown in table (13).

The unloaded and loaded computation times for each loop are shown in Tables (14) and (12), where "unloaded computation time refers to the first run of the software. If the software is already running, and *.mat* files are created and loaded into the workspace of MATLAB, requiring marked less computation time. However, this computation time depends on the features of the computer running the simulation.

IV. CONCLUSION

In this project, a software program was created for the identification of dynamic parameters for a collaborative robot. For the identification procedure, model derivation, optimization experiments, data collection, parameter estimation, and validation models were created in MATLAB. A user must then input the DH parameter of the robot and create an exciting trajectory using MATLAB or V-Rep. The developed program was checked using a collaborative robot called 7-DOF Kinova Gen2 and has been analyzed in detail.

The Newton-Euler formulation was used to model robot dynamics. To develop an identification software, users will use various types of robots, and perspective preference plays a role in this decision. The main benefit of this method is telling the user which parameters can be identified, non-identifiable, and linear parameters.

The main purpose of the proposed designed software is to support a user in the estimation of a robot's parameters in a few steps. The software runs under the Linux operating system and takes advantage of the MATLAB, ROS, and V-Rep environments. MATLAB is used as a computing tool, while ROS is responsible for data collection. To facilitate the design of exciting trajectories, V-REP software is used. The benefit of the software is that it uses V-Rep, which aids in system safety management. Before executing a real robot trajectory, the exciting trajectory can be executed in the V-Rep simulation, which reduces the chance of collision or damage to any mechanical structure of the real robot.

The benefit of this software is the capacity to anticipate required torque, improving the efficiency of control calculations that are dependent on the precomputed torque. Then, the produced module could actually be applied to various techniques of identification and analysis. ROS improves controllability and analysis options, and V-Rep helps to improve guidance and provides the ability to virtually verify the investigation without facing any danger of harm to the executive.

This identification software did not create Kinova Gen2 or Kuka LWR4+. First, this tool was created based on a novel methodology and then tested with a Kinova Gen2 robot and a Kuka LWR4+ robot. The experimental results showed that the algorithm operates effectively for a collaborative robot, which was the primary target of this study: the identification software must operate with collaborative robots that were not specifically created for them.

ACKNOWLEDGMENT

The data used to support the findings of this study are available from the corresponding author upon request.

REFERENCES

[1] J. Swevers, W. Verdonck, and J. D. Schutter, "Dynamic model identification for industrial robots," *IEEE Control Syst.*, vol. 27, no. 5, pp. 58–71, Oct. 2007.

[2] C. G. Atkeson, C. H. An, and J. M. Hollerbach, "Estimation of inertial parameters of manipulator loads and links," *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 101–119, Sep. 1986.

[3] J. Swevers, C. Ganseman, D. B. Tukel, J. de Schutter, and H. Van Brussel, "Optimal robot excitation and identification," *IEEE Trans. Robot. Autom.*, vol. 13, no. 5, pp. 730–740, Oct. 1997.

[4] M. Ciszewski, Ł. Mitka, T. Buratowski, and M. Giergiel, "Modeling and simulation of a tracked mobile inspection robot in MATLAB and V-REP software," *J. Autom., Mobile Robot. Intell. Syst.*, vol. 11, no. 2, pp. 5–11, Jun. 2017.

[5] M. Safeea and P. Neto, "KUKA sunrise toolbox: Interfacing collaborative robots with Matlab," *IEEE Robot. Autom. Mag.*, vol. 26, no. 1, pp. 91–96, Mar. 2019.

[6] P. I. Corke, "A simple and systematic approach to assigning Denavit–Hartenberg parameters," *IEEE Trans. Robot.*, vol. 23, no. 3, pp. 590–594, Jun. 2007.

[7] A. Campeau-Lecours, H. Lamontagne, S. Latour, P. Fauteux, V. Maheu, F. Boucher, C. Deguire, and L.-J. C. L'Ecuyer, "Kinova modular robot arms for service robotics applications," in *Rapid Automation: Concepts, Methodologies, Tools, and Applications*. Hershey, PA, USA: IGI Global, 2019, pp. 693–719.

[8] Z. H. Khan, A. Siddique, and C. W. Lee, "Robotics utilization for healthcare digitization in global COVID-19 management," *Int. J. Environ. Res. Public Health*, vol. 17, no. 11, p. 3819, May 2020.

[9] G. Schreiber, A. Stemmer, and R. Bischoff, "The fast research interface for the KUKA lightweight robot," in *Proc. IEEE Workshop Innov. Robot Control Archit. Demanding (Research) Appl. How Modify Enhance Commercial Controllers (ICRA)*. Princeton, NJ, USA: Citeseer, 2010, pp. 15–21.

[10] K. Lee, Y. Choi, and J. Park, "Inverse optimal design for position control of a quadrotor," *Appl. Sci.*, vol. 7, no. 9, p. 907, Sep. 2017.
[11] P. Zamora-Ortiz, J. Carral-Alvaro, Á. Valera, J. L. Pulloquinga, R. J. Escarabajal, and V. Mata, "Identification of inertial parameters for position and force control of surgical assistance robots," *Mathematics*, vol. 9, no. 7, p. 773, Apr. 2021.
[12] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano, and A. D. Luca, "Dynamic identification of the Franka Emika panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4147–4154, Oct. 2019.
[13] P. Chang and T. Padır, "Model-based manipulation of linear flexible objects: Task automation in simulation and real world," *Machines*, vol. 8, no. 3, p. 46, Aug. 2020.
[14] L. Sciavicco and B. Siciliano, "Modelling and control of robot manipulators," *Ind. Robot, Int. J.*, vol. 25, no. 1, p. 73, 1998.
[15] V. Mata, F. Benimeli, N. Farhat, and A. Valera, "Dynamic parameter identification in industrial robots considering physical feasibility," *Adv. Robot.*, vol. 19, no. 1, pp. 101–119, Jan. 2005.
[16] K. Shojaei, A. Kazemy, and A. Chatraei, "An observer-based neural adaptive PID^2 controller for robot manipulators including motor dynamics with a prescribed performance," *IEEE/ASME Trans. Mechatronics*, vol. 26, no. 3, pp. 1689–1699, Jun. 2021.
[17] J. W. Demmel, *Applied Numerical Linear Algebra*. Philadelphia, PA, USA: SIAM, 1997.
[18] M. Gautier, "Dynamic identification of robots with power model," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 3, Apr. 1997, pp. 1922–1927.
[19] M. Rahmani, H. Komijani, and M. H. Rahman, "New sliding mode control of 2-DOF robot manipulator based on extended grey wolf optimizer," *Int. J. Control, Autom. Syst.*, vol. 18, pp. 1572–1580, Jan. 2020.
[20] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE J. Robot. Autom.*, vol. RA-3, no. 1, pp. 43–53, Feb. 1987.
[21] Y. Chen, X. Luo, B. Han, Q. Luo, and L. Qiao, "Model predictive control with integral compensation for motion control of robot manipulator in joint and task spaces," *IEEE Access*, vol. 8, pp. 107063–107075, 2020.
[22] G. Antonelli, F. Caccavale, and P. Chiacchio, "A systematic procedure for the identification of dynamic parameters of robot manipulators," *Robotica*, vol. 17, no. 4, pp. 427–435, Jul. 1999.
[23] J. Jia, M. Zhang, X. Zang, H. Zhang, and J. Zhao, "Dynamic parameter identification for a manipulator with joint torque sensors based on an improved experimental design," *Sensors*, vol. 19, no. 10, p. 2248, May 2019.
[24] N. X. Quynh, W. Y. Nan, and V. T. Yen, "Design of a robust adaptive sliding mode control using recurrent fuzzy wavelet functional link neural networks for industrial robot manipulator with dead zone," *Intell. Service Robot.*, vol. 13, no. 2, pp. 219–233, Apr. 2020.
[25] M. Gautier and W. Khalil, "Exciting trajectories for the identification of base inertial parameters of robots," *Int. J. Robot. Res.*, vol. 11, no. 4, pp. 362–375, Aug. 1992.
[26] G. W. Stewart, *Matrix Perturbation Theory*. Boston, MA, USA: Academic, 1990.
[27] M. Madonna, L. Monica, S. Anastasi, and M. D. Nardo, "Evolution of cognitive demand in the human–machine interaction integrated with industry 4.0 technologies," *Wit Trans. Built Environ.*, vol. 189, pp. 13–19, Nov. 2019.
[28] R. R. Shamshiri, I. A. Hameed, M. Karkee, and C. Weltzien, "Robotic harvesting of fruiting vegetables: A simulation approach in V-REP, ROS and MATLAB," in *Automation in Agriculture—Securing Food Supplies for Future Generations*. London, U.K.: IntechOpen, 2018.
[29] M. Freese, S. Singh, F. Ozaki, and N. Matsuhira, "Virtual robot experimentation platform V-REP: A versatile 3D robot simulator," in *Proc. Int. Conf. Simulation, Modeling, Program. Auto. Robots*. Cham, Switzerland: Springer, 2010, pp. 51–62.
[30] Z. B. Rivera, M. C. De Simone, and D. Guida, "Unmanned ground vehicle modelling in Gazebo/ROS-based environments," *Machines*, vol. 7, no. 2, p. 42, Jun. 2019.
[31] G. Farias, E. Torres, E. Fabregas, H. Vargas, S. Dormido-Canto, and S. Dormido, "Navigation control of the khepera IV model with OpenCV in V-REP simulator," in *Proc. IEEE Int. Conf. Automat./23th Congr. Chilean Assoc. Autom. Control (ICA-ACCA)*, Oct. 2018, pp. 1–6.
[32] M. A. Freese. (2011). *V-REP*. [Online]. Available: http://www.coppeliarobotics.com
[33] J. Batista, D. Souza, L. dos Reis, A. Barbosa, and R. Araújo, "Dynamic model and inverse kinematic identification of a 3-DOF manipulator using RLSPSO," *Sensors*, vol. 20, no. 2, p. 416, Jan. 2020.
[34] W. Chen, H. Ma, D. Yu, and H. Zhang, "SVD-based technique for interference cancellation and noise reduction in NMR measurement of time-dependent magnetic fields," *Sensors*, vol. 16, no. 3, p. 323, Mar. 2016.
[35] T. Piatkowski, "GMS friction model approximation," *Mechanism Mach. Theory*, vol. 75, pp. 1–11, May 2014.
[36] K. Yoshida and W. Khalil, "Verification of the positive definiteness of the inertial matrix of manipulators using base inertial parameters," *Int. J. Robot. Res.*, vol. 19, no. 5, pp. 498–510, May 2000.
[37] K. Miura, S. Tokunaga, N. Ota, Y. Tange, and T. Azumi, "Autoware toolbox: MATLAB/Simulink benchmark suite for ROS-based self-driving software platform," in *Proc. 30th Int. Workshop Rapid Syst. Prototyping (RSP)*, Oct. 2019, pp. 8–14.
[38] L. Nardi, B. Bodin, M. Z. Zia, J. Mawer, A. Nisbet, P. H. J. Kelly, A. J. Davison, M. Lujan, M. F. P. O'Boyle, G. Riley, N. Topham, and S. Furber, "Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 5783–5790.

ÖMER AYVACI received the master's degree in automatic control and robotics engineering from the Poznań University of Technology, in 2016, where he is currently pursuing the Ph.D. degree in automatic control and robotics engineering. His current research interest includes identification methods in robotics.

PAWEŁ SZULCZYŃSKI received the Ph.D. degree in automatic control and robotics engineering from the Poznań University of Technology, in 2012. He is currently working as an Academic Member of the Poznań University of Technology.

MARCIN KIEŁCZEWSKI received the M.Sc. and Ph.D. degrees in automatics and robotics from the Poznań University of Technology, Poland, in 2000 and 2010, respectively. He is currently an Assistant Professor with the Institute of Automatic Control and Robotics, PUT, where he teaches and conducts research. His current research interests include image processing and applications of vision systems in robotics and automation, especially in industrial manipulators and mobile robots.