2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9635859

Kalibrot: A Simple-To-Use Matlab Package for Robot Kinematic Calibration

Francesco Cursi*,1,2, Weibang Bai1, *Member, IEEE*, Petar Kormushev2, *Member, IEEE*

Abstract— **Robot modelling is an essential part to properly understand how a robotic system moves and how to control it. The kinematic model of a robot is usually obtained by using Denavit-Hartenberg convention, which relies on a set of parameters to describe the end-effector pose in a Cartesian space. These parameters are assigned based on geometrical considerations of the robotic structure, however, the assigned values may be inaccurate. The purpose of robot kinematic calibration is therefore to find optimal parameters which improve the accuracy of the robot model. In this work we present Kalibrot, an open source Matlab package for robot kinematic calibration. Kalibrot has been designed to simplify robot calibration and easily assess the calibration results. Beside computing the optimal parameters, Kalibrot provides a visualization layer showing the values of the calibrated parameters, what parameters can be identified, and the calibrated robotic structure. The capabilities of the package are here shown through simulated and real world experiments.**

I. Introduction

Robot kinematics allows the structure of a robot to be modelled, mapping the joint values to the operational space, usually a six-dimensional Cartesian space. Kinematic modelling is of utmost importance because it is the foundation of kinematic control. If the model of the robot is known, proper control strategies can be implemented, for instance, to have trajectory tracking. Having good kinematic models allows to properly control the robotic system, without requiring complex compensation strategies, which is essential especially in cases where it is not possible to rely on external sensors to compensate for positioning errors.

There exist a large variety of robotic structures, such as rigid-link articulated robots, flexible-link robots, continuum robots, soft robots [1]–[5]. Depending on the structure, different modelling techniques exist. Articulated robots with rigid links are usually modelled by using Denavit-Hartenberg (DH) convention [6]. This is a simple and effective method which allows to characterize the pose of a link of a robot with respect to the previous link by using a set of parameters (DH parameters) related to the link geometry and the joint type (revolute or prismatic) as in Figure 1.

The DH parameters, however, are usually assigned based on considerations from the structure of the link and their values may not be accurate due to errors such as manufacturing inaccuracies or joints offsets. This, in turn, leads to

*Corresponding author. This work was supported by EPSRC project EP/P012779/1.

1 Hamlyn Centre, Imperial College London, Exhibition Road, London, UK.
2 Robot Intelligence Lab, Imperial College London, Exhibition Road, London, UK.
Email: [f.cursi17, wbbai, p.kormushev]@imperial.ac.uk

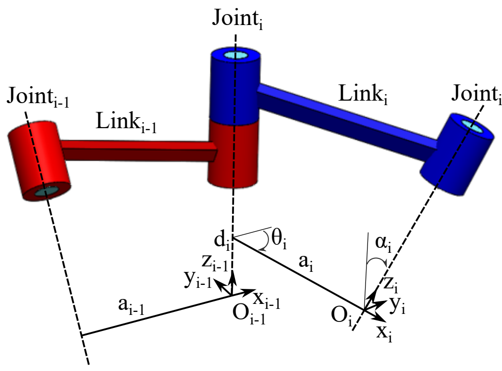


Fig. 1: Denavit-Hartenberg parameters representation [6].

unwanted positioning inaccuracies of the robot. The purpose of kinematic calibration is thus to reduce the modelling errors in the system [7]. The calibration problem can therefore be formulated as an optimization problem where the cost function is the error between the measured robot end-effector pose and the modelled one, and with the DH parameters being the optimization variables [6], [7].

The first step for an optimal calibration is data acquisition. Many different sensors have been adopted in order to collect the position data of the end-effector, varying from laser trackers [8], optical trackers [9], stereo-vision systems [10]. Other approaches consist in constraining the end-effector to a specific movement, where joint angle values are collected through a probe triggering [11].

After data collection, the identification procedure is carried out. Different methods exist in literature such as two step nonlinear optimization [12], Unscented Kalman Filter [13], product of exponential formula [14]. However, the most typical approach is based on a linearization of the nonlinear cost function and then iteratively solving the linear problem in a least squares sense until convergence [6], [15].

Thus far, no software is available online to allow performing robot calibration easily, especially for inexperienced users. Moreover, calibration data and exact parameters for widely used robots are very hard to find online. Therefore, this work aims to providing a simple-to-use open-source Matlab package for robot kinematic calibration, named Kalibrot (available at *https://github.com/cursi36/Kalibrot*). Additionally, we would like to provide an open-source database to store calibration datasets of commonly used manipulators, in order for the community to have easier access to this information and be able to build more accurate simulation. Currently, Kalibrot focuses only on DH parameters optimization and it allows the user to set up and solve the calibration problem very easily, by using the implemented functions. Two different solvers are implemented to solve the calibration problem. Since the solution of the optimization problem for parameter identification requires computing the derivatives of the cost function with respect to the DH parameters, Kalibrot performs automatic differentiation to compute the derivatives of the cost function for the DH parameters optimization analytically. Beside the calibrated DH parameters, Kalibrot also provides other useful information about the calibration, such as the number of parameters that can be identified and the observability measures for assessing the accuracy of the data. Moreover, the visualization layer allows the user to plot the calibration results and visualize the calibrated robot kinematic structure. However, the capabilities of Kalibrot can be further improved by adding other functions or plugins and we would encourage the community to contribute to expand its features.

The paper is thus organized as follows. Section II briefly describes the problem of robot kinematic calibration. Section III describes the proposed package Kalibrot for kinematic calibration, presenting its architecture and its functionalities. Section IV shows the results both in simulated and real world experiments using Kalibrot. Finally, conclusions are drawn in Section V.

II. Robot Kinematic Calibration

In this section the kinematic calibration problem is formulated, presenting the robot kinematic modelling and the optimization problem for parameter identification.

A. Robot Kinematic Modelling

Robot kinematic calibration is a necessary step to have accurate robot models, and, therefore, to implement successful control strategies. Given a certain robot model, the purpose is to find the optimal modelling parameters that allow the model to be as accurate as possible. Therefore, robot kinematic calibration requires an a priori known model of the robot.

Different methods exist to build kinematic models of robots, yet, one of the most widespread is based on Denavit-Hartenberg convention. According to this method, the transformation matrix ${}^{i-1}\boldsymbol{T}_i \in \mathbb{R}^{4\times4}$, relating the pose of a link i to the preceding one, can be described by means of 4 parameters $d_i, \theta_i, a_i, \alpha_i$ and the joint value q_i as

$$ {}^{i-1}\boldsymbol{T}_i = \begin{bmatrix} \cos(\tilde{\theta}_i) & -\sin(\tilde{\theta}_i)\cos(\alpha_i) & \sin(\tilde{\theta}_i)\sin(\alpha_i) & a_i\cos(\tilde{\theta}_i) \\ \sin(\tilde{\theta}_i) & \cos(\tilde{\theta}_i)\cos(\alpha_i) & -\cos(\tilde{\theta}_i)\sin(\alpha_i) & a_i\sin(\tilde{\theta}_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & \tilde{d}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1) $$

where

$$ \begin{cases} \tilde{d}_i = d_i + q_i, & \text{if joint is prismatic} \\ \tilde{\theta}_i = \theta_i + q_i, & \text{if joint is revolute} \end{cases} . \quad (2) $$

d_i is the distance between two consecutive joints along the the current joint axis, θ_i the tilting between the two joints about the current joint axis, a_i the link's length, and α_i the angle between the two joint axes (Figure 1). Following the chain rule [6], the robot end-effector pose in a desired world frame $\{RF_0\}$ can be expressed as

$$ \begin{aligned} {}^{0}\boldsymbol{T}_{ee}(\boldsymbol{q},\boldsymbol{x}) &= \begin{bmatrix} {}^{0}\boldsymbol{R}_{ee} & {}^{0}\boldsymbol{r}_{ee} \\ 0 & 1 \end{bmatrix} \\ &= {}^{0}\boldsymbol{T}_1(q_1,\boldsymbol{x}_1)\ {}^{1}\boldsymbol{T}_2(q_2,\boldsymbol{x}_2)\dots\ {}^{n-1}\boldsymbol{T}_{ee}(q_n,\boldsymbol{x}_n) \end{aligned} \quad (3) $$

with ${}^{0}\boldsymbol{R}_{ee} \in \mathbb{R}^{3\times3}$ being the rotation matrix of the end-effector, ${}^{0}\boldsymbol{r}_{ee} \in \mathbb{R}^3$ the end-effector position, $\boldsymbol{x}_i = \begin{bmatrix} d_i & \theta_i & a_i & \alpha_i \end{bmatrix}^T$, $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_n \end{bmatrix}^T \in \mathbb{R}^{4n}$, and $\boldsymbol{q} \in \mathbb{R}^n$, being n the number of joints. Thanks to this representation, it is then possible to extract the end-effector position and orientation and express it as a nonlinear function of the joint values and the DH parameters as

$$ \boldsymbol{P} = \begin{bmatrix} {}^{0}\boldsymbol{r}_{ee} \\ {}^{0}\boldsymbol{Q}_{ee} \end{bmatrix} = \boldsymbol{P}(\boldsymbol{q},\boldsymbol{x}) \in \mathbb{R}^7 , \quad (4) $$

where ${}^{0}\boldsymbol{Q}_{ee} \in \mathbb{R}^4$ represents the quaternion of the orientation of the end-effector with respect to the base frame. Different representations for the orientation can also be used.

B. Optimal Parameter Identification

To identify the optimal parameters and correct the actual robot model, the kinematic calibration problem can be formulated as a nonlinear optimization problem as

$$ \boldsymbol{x} = \arg\min_{\boldsymbol{x}} \frac{1}{2}||\tilde{\boldsymbol{P}} - \boldsymbol{P}(\boldsymbol{q},\boldsymbol{x})||^2 , \quad (5) $$

where $\tilde{\boldsymbol{P}} \in \mathbb{R}^{7M}$ is a vector resulting from stacking all the position and orientation measurements at each robot configuration. M is the total number of measurements. However, to simplify the problem, it is usual to consider a linearization of the model and then iteratively solve the optimization problem until convergence [6] as in Algorithm1.

Algorithm 1 Linearized optimization algorithm.

```
1: function $\boldsymbol{x} \leftarrow$ ITERATIVEOPT($\boldsymbol{x}_0, \tilde{\boldsymbol{P}}, \boldsymbol{q}$)
2:     $\boldsymbol{x} = \boldsymbol{x}_0$
3:     while $err > Err$ do
4:         $\mathbf{P} \sim \boldsymbol{P}(\boldsymbol{q},\boldsymbol{x}) + \boldsymbol{D}\Delta\boldsymbol{x}$        ▷ Linearize model
5:         $\Delta\boldsymbol{x} = \arg\min_{\Delta\boldsymbol{x}} \frac{1}{2}||\tilde{\boldsymbol{P}} - \boldsymbol{P}(\boldsymbol{q},\boldsymbol{x}) - \boldsymbol{D}\Delta\boldsymbol{x}||^2$
6:         $\boldsymbol{x} = \boldsymbol{x} + \Delta\boldsymbol{x}$
7:     end while
8: end function
```

The matrix $\boldsymbol{D}$ is the Jacobian of the position and orienatation with respect to all the DH parameters $\mathbf{x}$. This matrix can be computed numerically, yet it leads to high computational efforts and lower accuracy in the solution. Moreover, typically the problem is solved using simple pseudoinversion, leading to $\Delta\boldsymbol{x} = \boldsymbol{D}^{\dagger}(\tilde{\boldsymbol{P}} - \boldsymbol{P})$ at each iteration. However, simple pseudinversion may not lead to good solutions, since it doesn't take into account possible constraints on the values of DH parameters.

III. Kalibrot: Kinematic Calibration Package

In this section, Kalibrot functionalities are presented, describing the underlying architecture, the procedure for the analytical computation of the derivatives of the calibration

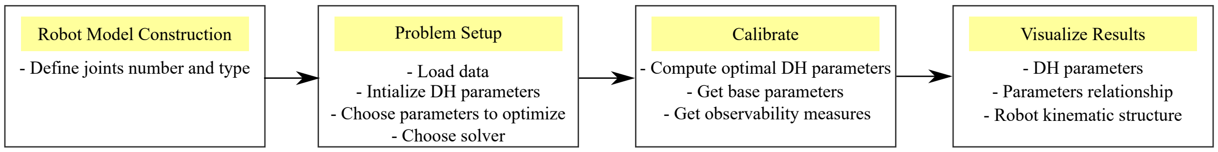

Robot Model Construction
- Define joints number and type
Problem Setup
- Load data
- Intialize DH parameters
- Choose parameters to optimize
- Choose solver
Calibrate
- Compute optimal DH parameters
- Get base parameters
- Get observability measures
Visualize Results
- DH parameters
- Parameters relationship
- Robot kinematic structure

Fig. 2: Kalibrot workflow: create the robot model; setup the calibration problem; perform the calibration; visualize the calibration results.

cost function, and the implemented formulations of the optimization problem for DH parameters identification.

A. Kalibrot Workflow

Kalibrot was designed to be simple to use and allow to setup and solve the calibration problem very easily.

Figure 2 shows the workflow of the package. First, the robot model needs to be built. This is done by specifying the number of links and their types (if prismatic or revolute). The *RobotKinematics* class then creates the robot model. This class also implements the functions for computing the robot kinematics. After the robot model is created, the calibration problem is set up. The data, consisting in a set of joint values and corresponding poses (expressed in terms of 3D Cartesian position and quaternion), is loaded and the DH parameters are initialized with some initial guesses. Some bounds on the values of the DH parameters can be imposed, depending on what solver is used, as described in the following. The user can select what parameters to optimize for and what motion component to take into account for the calibration (e.g. considering only the 3D position, without any regard for the orientation). Then the solver is chosen and the calibration run. At the end of the calibration, the user can choose to enable the visualization of the results. A snippet of the code for using Kalibrot is shown in the complementary video.

B. Analytical Derivatives Computation

In order to compute the derivatives of the position and orientation vectors with respect to the DH parameters, analytical methods are preferred to numerical ones because they do not rely on approximations and because they are faster. Symbolic toolboxes, like Matlab, can be used to retrieve $\boldsymbol{P}(\boldsymbol{q}, \boldsymbol{x})$ symbolically as a function of the DH parameters. This allows to also directly find the Jacobian matrix $\boldsymbol{D}$ very easily. Nevertheless, the computation is not very fast. The approach implemented in Kalibrot allows to easily compute the matrix of derivatives for any given robotic structure, simply defined by the DH parameters. For the sake of brevity, the whole derivation is not reported in the manuscript.

For the analytical computation of the derivatives, an iterative method can be employed. As a matter of fact, from (3) the position vector of the end-effector for an n degree of freedom (DOF) robot can be computed iteratively as:

$$\begin{aligned}
{}^{0}\boldsymbol{r}_{ee} &= {}^{0}\boldsymbol{r}_1 + {}^{0}\boldsymbol{R}_1\,{}^{1}\boldsymbol{r}_{ee}\\
{}^{1}\boldsymbol{r}_{ee} &= {}^{1}\boldsymbol{r}_2 + {}^{1}\boldsymbol{R}_2\,{}^{2}\boldsymbol{r}_{ee}\\
&\vdots\\
{}^{n-1}\boldsymbol{r}_{ee} &= {}^{n-1}\boldsymbol{r}_n + {}^{n-1}\boldsymbol{R}_n\,{}^{n}\boldsymbol{r}_{ee}\ ,
\end{aligned} \tag{6}$$

where ${}^{i-1}\boldsymbol{r}_i, {}^{i-1}\boldsymbol{R}_i$ are the position and orientation matrix of joint i with respect to the previous one, ${}^{i}\boldsymbol{r}_{ee}$ is the position of the end effector with respect to the joint i.

Therefore, for a given link i, the differential of the position can be written as

$${}^{i-1}d\boldsymbol{r}_{ee} = {}^{i-1}d\boldsymbol{r}_i + {}^{i-1}d\boldsymbol{R}_i\,{}^{i}\boldsymbol{r}_{ee} + {}^{i-1}\boldsymbol{R}_i\,{}^{i}d\boldsymbol{r}_{ee} \tag{7}$$

However, from (1), the differentials ${}^{i-1}d\boldsymbol{r}_i$ and ${}^{i-1}d\boldsymbol{R}_i$ depend only on the link's DH parameters $\boldsymbol{x}_i = \begin{bmatrix} d_i & \theta_i & a_i & \alpha_i \end{bmatrix}^T$ and can be easily computed in closed form. Eventually it is possible to iteratively compute the end-effector differential in the base frame as

$$\begin{aligned}
{}^{0}d\boldsymbol{r}_{ee} &= \boldsymbol{D}_{p_1}d\boldsymbol{x}_1 + {}^{0}\boldsymbol{R}_1\boldsymbol{D}_{p_2}d\boldsymbol{x}_2 + \cdots + {}^{0}\boldsymbol{R}_{n-1}\boldsymbol{D}_{p_n}d\boldsymbol{x}_n\\
&= \boldsymbol{D}_p d\boldsymbol{x}
\end{aligned} \tag{8}$$

with each $\boldsymbol{D}_{p_i}$ being a matrix of derivatives of the position and rotation matrix for each joint, and $\boldsymbol{D}_p \in \mathbb{R}^{3\times 4n}$.

With regards to the orientation, the quaternion of the end-effector frame can be expressed as

$${}^{0}\boldsymbol{Q}_{ee} = {}^{0}\boldsymbol{Q}_1 * {}^{1}\boldsymbol{Q}_2 * \cdots * {}^{n-1}\boldsymbol{Q}_n\ , \tag{9}$$

where $*$ is the Hamilton or quaternion product [6], and ${}^{i-1}\boldsymbol{Q}_i = \{\eta_i, \boldsymbol{\xi}_i\}$ describes the orientation of ${}^{i-1}\boldsymbol{T}_i$, with $\eta_i \in \mathbb{R}$ and $\boldsymbol{\xi}_i = \begin{bmatrix} \xi_{i,x} & \xi_{i,y} & \xi_{i,z} \end{bmatrix}^T \in \mathbb{R}^3$.

In order to compute $\{\eta_i, \boldsymbol{\xi}_i\}$, one must first obtain ${}^{i-1}\boldsymbol{R}_i$. Since this rotation matrix depends only on the actual link's DH parameters $\boldsymbol{x}_i = \begin{bmatrix} d_i & \theta_i & a_i & \alpha_i \end{bmatrix}^T$, the differential of the quaternion can be computed as

$$\begin{aligned}
&{}^{i-1}d\boldsymbol{Q}_i = \{d\eta_i, d\boldsymbol{\xi}_i\}\\
&\text{with}\\
&d\eta_i = \boldsymbol{D}_{\eta_i}d\boldsymbol{x}_i \in \mathbb{R}\\
&d\boldsymbol{\xi}_i = \boldsymbol{D}_{\xi_i}d\boldsymbol{x}_i \in \mathbb{R}^3
\end{aligned} \tag{10}$$

These derivatives can be easily computed analytically.

From (9) and from the definition of the quaternion product [6], for a given link i it is then possible to write

$$\begin{aligned}
{}^{0}\eta_i &= {}^{0}\eta_{i-1}\eta_i - {}^{\mathbf{0}}\mathbf{\xi}_{\mathbf{i-1}}^{\mathbf{T}}\mathbf{\xi}_{\mathbf{i}}\\
{}^{0}\boldsymbol{\xi}_i &= {}^{0}\eta_{i-1}\boldsymbol{\xi}_i + \eta_i\,{}^{0}\boldsymbol{\xi}_{i-1} + {}^{0}\boldsymbol{\xi}_{i-1}\times\boldsymbol{\xi}_i
\end{aligned}\ , \tag{11}$$

which yields the following results

$$\begin{aligned}
{}^{0}d\eta_i &= {}^{0}d\eta_{i-1}\eta_i + {}^{0}\eta_{i-1}d\eta_i - {}^{0}d\boldsymbol{\xi}_{i-1}^T\boldsymbol{\xi}_i - {}^{0}\boldsymbol{\xi}_{i-1}^T d\boldsymbol{\xi}_i\\
{}^{0}d\boldsymbol{\xi}_i &= {}^{0}d\eta_{i-1}\boldsymbol{\xi}_i + {}^{0}\eta_{i-1}d\boldsymbol{\xi}_i + d\eta_i\,{}^{0}\boldsymbol{\xi}_{i-1} +\\
&\quad \eta_i\,{}^{0}d\boldsymbol{\xi}_{i-1} + {}^{0}d\boldsymbol{\xi}_{i-1}\times\boldsymbol{\xi}_i + {}^{0}\boldsymbol{\xi}_{i-1}\times d\boldsymbol{\xi}_i
\end{aligned}\ . \tag{12}$$

Since ${}^0d\eta_{i-1}$ and ${}^0d\boldsymbol{\xi}_{i-1}$ depend only on the DH parameters of all the preceding links $\boldsymbol{x}_1 \dots \boldsymbol{x}_{i-1}$, it is then possible to obtain

$$
{}^0d\boldsymbol{Q}_i = \begin{bmatrix} {}^0d\eta_i \\ {}^0d\boldsymbol{\xi}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}_{\eta_{i-1}} & \boldsymbol{D}_{\eta_i} \\ \boldsymbol{D}_{\xi_{i-1}} & \boldsymbol{D}_{\xi_i} \end{bmatrix} \begin{bmatrix} d\boldsymbol{x}_1 \\ d\boldsymbol{x}_2 \\ \vdots \\ d\boldsymbol{x}_i \end{bmatrix} . \quad (13)
$$

Finally, the end-effector quaternion differential can be obtained as

$$
{}^0d\boldsymbol{Q}_{ee} = \boldsymbol{D}_{or} d\boldsymbol{x} \ , \quad (14)
$$

with $\boldsymbol{D}_{or} \in \mathbb{R}^{4\times 4n}$. From (8) and (14) the pose Jacobian matrix with respect to the DH parameters, for each data point $m = 1 \dots M$, is computed as $\boldsymbol{D}_m = \begin{bmatrix} \boldsymbol{D}_p \\ \boldsymbol{D}_{or} \end{bmatrix}$. These matrices are then stacked together for solving the calibration problem, to obtain

$$
\boldsymbol{D} = \begin{bmatrix} \boldsymbol{D}_1 \\ \vdots \\ \boldsymbol{D}_M \end{bmatrix} \in \mathbb{R}^{7M\times 4n} \ . \quad (15)
$$

A similar approach is to consider each DH parameter as an active joint [16], but this would require to iterate through the robotic kinematic chain and add the DH parameters as new joints, resulting in less flexibility.

C. Kalibrot Calibration Solvers

The ease of use of Kalibrot resides also in the possibility to choose different solvers and methods for the calibration problem. Currently two different approaches are implemented:

- *Pinv*: based on standard pseudoinversion (as in Algorithm 1);
- *QP*: constrained Quadratic Programming solver.

Other methods can be easily implemented in the Kalibrot framework as additional plugins.

For the *Pinv* method, the optimal DH parameters are obtained by iteratively solving the following optimization problem

$$
\begin{aligned}
\Delta\boldsymbol{x}_j = &\arg\min_{\Delta\boldsymbol{x}_j} \frac{1}{2} ||\boldsymbol{W}(\tilde{\boldsymbol{P}} - \boldsymbol{P}(\boldsymbol{q}, \boldsymbol{x}_j)) - \boldsymbol{W}\boldsymbol{D}_j\boldsymbol{W}_p\Delta\boldsymbol{x}_j||^2 \\
&+ \lambda||\Delta\boldsymbol{x}_j||^2
\end{aligned} , \quad (16)
$$

which results in the damped least squares solution $\Delta\boldsymbol{x}_j = \boldsymbol{H}^{-1}\boldsymbol{g}$, where $\boldsymbol{H} = \boldsymbol{W}_p^T\boldsymbol{D}_j^T\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{D}_j\boldsymbol{W}_p$ and $\boldsymbol{g} = \boldsymbol{W}_p^T\boldsymbol{D}_j^T\boldsymbol{W}^T\boldsymbol{W}(\tilde{\boldsymbol{P}} - \boldsymbol{P}(\boldsymbol{q}, \boldsymbol{x}_j))$. The diagonal matrix $\mathbf{W_p} \in \mathbb{R}^{4n\times 4n}$ is a matrix of zeros and ones used to choose which parameters to optimize for. $\boldsymbol{D}_j$ is the matrix of derivatives (15) computed with the current estimates of the DH parameters at iteration j. The matrix $\mathbf{W}$, instead, is used to assign different weights to the data points or to choose which Cartesian component to optimize for. The regularization term $\lambda||\Delta\boldsymbol{x}_j||^2$ is added in order to limit the error between the linearized model and the actual model. As a matter of fact, if the resulting increment of the DH parameter is too big, the model linearization fails. The parameter λ is updated in a Levenberg-Marquardt algorithm [17] fashion. An initial value is provided by the user. If the current DH parameters update leads to a worse value of the cost function, then the solution is discarded and λ increased; otherwise the DH parameters are updated and λ decreased. The process continues until the cost functions goes below a certain threshold.

Obtaining the increment of the DH parameters at each iteration simply by using the pseudoinversion may not lead to a feasible solution. As a matter of fact, this approach will only try to minimize the error between the actual estimated pose and the measured one, yielding optimal DH parameters that may be unnatural (i.e. negative or very large a_i or d_i). This occurrence is much more likely if the initial DH parameters are far away from the optimal ones or if the acquired data is not good enough.

In order to overcome this problem, the *QP* solver can be employed, which solves, at each iteration j, a constrained quadratic programming problem (QP) as

$$
\begin{aligned}
\Delta\boldsymbol{x}_j = &\arg\min_{\Delta\boldsymbol{x}_j} \frac{1}{2} ||\boldsymbol{W}(\tilde{\boldsymbol{P}} - \boldsymbol{P}(\boldsymbol{q}, \boldsymbol{x}_j)) - \boldsymbol{W}\boldsymbol{D}_j\boldsymbol{W}_p\Delta\boldsymbol{x}_j||^2 \\
&+ \lambda||\Delta\boldsymbol{x}_j||^2 \\
&\text{s.t} \ \ \boldsymbol{lb}_j \le \Delta\boldsymbol{x}_j \le \boldsymbol{ub}_j
\end{aligned} . \quad (17)
$$

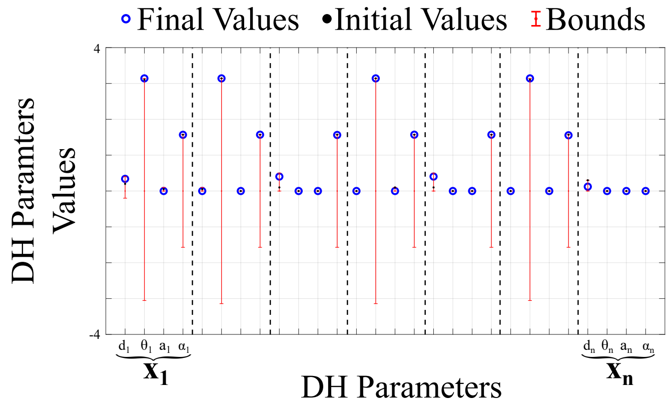
The bounds $\boldsymbol{lb}_j, \boldsymbol{ub}_j$ are updated at each iteration in order to guarantee that the DH parameters always lie between the imposed bounds $\boldsymbol{x}_{min}, \boldsymbol{x}_{max}$. The bounds are therefore computed as $\boldsymbol{lb}_j = \boldsymbol{x}_{min} - \boldsymbol{x}_j$ and $\boldsymbol{ub}_j = \boldsymbol{x}_{max} - \boldsymbol{x}_j$. The same adaptation of λ as in the *Pinv* solver is used and the solver stops when the cost function is below the imposed threshold.

D. Additional Outputs

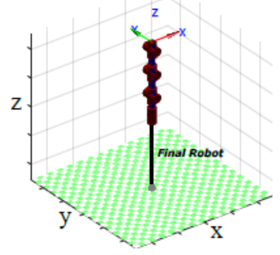
Beside the DH parameters estimation, Kalibrot provides additional information about the calibration procedure: the base parameters and some observability measures. Both the base parameters and the observability measures depend on the Jacobian matrix $\mathbf{D}$ (15), and, because of the nonlinear kinematic model and the iterative linear approximation, they may change from one iteration to the next one. Therefore, the solvers output the base parameters and the observability measures at each iteration.

1) Base Parameters: The possible rank deficiency in the Jacobian matrix $\boldsymbol{D}$ leads to some parameters not being identifiable or identifiable only in combination with other parameters. The base parameters correspond to these combinations and represent a minimum set of identifiable parameters [18]. To detect the base parameters Singular Value Decomposition (SVD) can be used.

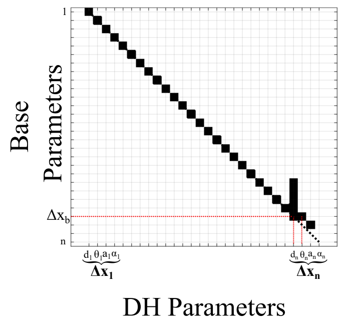
At each iteration j, given the Jacobian matrix $\boldsymbol{D}_j \in \mathbb{R}^{d\times N}$ for the whole set of datapoints, with $d = 7M$ and $N = 4n$, the SVD yields $\boldsymbol{D}_j = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$. If r is the rank of $\mathbf{D_j}$, then, because of rank deficiency, $\boldsymbol{V} = \begin{bmatrix} \boldsymbol{V}_1 & \boldsymbol{V}_2 \end{bmatrix}$, with $\boldsymbol{V}_2 \in \mathbb{R}^{N\times N-r}$, whose columns define the linear combinations of the columns of $\boldsymbol{D}_j$.



(a) DH parameters values



(b) Robotic structure



(c) Identification matrix

Fig. 3: Resulting plots after the calibration: 3a) the final and initial DH parameters estimations; 3b) the final estimated robot kinematic structure; 3c) the final identification matrix of the base parameters $\boldsymbol{K}$ (18).

Following the approach defined in [18], in order to find the base parameters, a permutation of the rows of $\boldsymbol{V}_2$ needs to be performed, such that $\boldsymbol{L}^T \Delta \boldsymbol{x}_j = \begin{bmatrix} \Delta \boldsymbol{x}_{j,1} \\ \Delta \boldsymbol{x}_{j,2} \end{bmatrix}$ and $\boldsymbol{L}^T \boldsymbol{V}_2 = \begin{bmatrix} \boldsymbol{V}_{1,1} \\ \boldsymbol{V}_{2,2} \end{bmatrix}$, with $\Delta \boldsymbol{x}_{j,2} \in \mathbb{R}^{N-r}$ being the parameters that can only be identified in linear combination with the others, $\boldsymbol{V}_{2,2} \in \mathbb{R}^{N-r \times N-r}$, and $\boldsymbol{L}$ being a permutation matrix that needs to be chosen such that $\boldsymbol{V}_{2,2}$ is regular. The set of base parameters $\Delta \boldsymbol{x}_{j,B} \in \mathbb{R}^N$ can then be retrieved as

$$\Delta \boldsymbol{x}_{j,B} = \begin{bmatrix} \Delta \boldsymbol{x}_{j,1} - \boldsymbol{V}_{1,1} - \boldsymbol{V}_{2,2}^{-1} \Delta \boldsymbol{x}_{j,2} \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{K}_j \Delta \boldsymbol{x}_j \ . \quad (18)$$

The identification matrix $\boldsymbol{K}_j$ is useful to assess which parameters can be identified, and which ones can be identified only in linear combination with other DH parameters.

2) Observability measures: The observability measures give information about the accuracy of the estimations. Kalibrot computes the following measures:

- *D-observability* defined as $o_1 = \sqrt{det(\boldsymbol{D}^T \boldsymbol{D})}$ [19];
- *Condition number* which computes $o_2 = \frac{\sigma_{max}}{\sigma_{min}}$, with $\sigma_{max}, \sigma_{min}$ being the minimum and maximum eigenvalues of $\boldsymbol{D}$ [20];
- *Minimum singular value* which defines $o_3 = \sigma_{min}$ [16].

E. Visualization

To make it easier for the user to analyze the results, Kalibrot also provides the possibility to visualize the outputs of the calibration. Figure 3 shows an example of the plots made by Kalibrot at the end of the calibration procedure. First of all, the resulting DH parameters are shown, along with the initial estimates. If *QP* solver is chosen, then also the imposed bounds are shown (Figure 3a). The obtained DH parameters are used to build and construct the robotic structure (Figure 3b). A specified configuration can be chosen by assigning the desired joint values. Finally, the final identification matrix $\boldsymbol{K}$ (18) for the identification of the base parameters is shown in a binary form. If a base parameter depends on a specific DH parameter, then the corresponding cell will be black, otherwise it will be white. If multiple cells are black on the same row, then that base parameter is a linear combination of those DH parameters. For instance, Figure 3c shows that d_1 can be identified independently of the others, whereas d_n cannot and it is in a linear combination with θ_n to define the base parameter Δx_b. The entries of $\boldsymbol{K}$ define the values of the coefficients for each linear combination.

IV. Results

In this section the results of using Kalibrot for robot calibration are shown, both in simulation and on a real robot.

A. Simulated DH Parameters Optimization

To prove the capabilities of Kalibrot, two simulated examples are proposed: kinematic calibration of a 3R planar robot, and of a 6DOF Stanford manipulator (Figure 4).

In order to generate the data, the robot kinematic models are built by assigning some desired DH parameters (Table Ia and Ib). As proposed in [21], [22] for workspace generation, all joint combinations are computed, considering that each joint can have two states: fixed or moving. In total, $2^n - 1$ combinations are obtained. At each combination, the moving joints are commanded a sinusoidal motion to move from the minimum to the maximum joint limits. Each sinusoid is discretized in 51 points, resulting in a total of $51 \cdot (2^n - 1)$ data points collected. For each commanded joint value, the corresponding end-effector pose is collected, expressing it in terms of 3D Cartesian position and quaternion. Gaussian noise, with zero mean and variance of $0.1m$ for the 3D position and $0.05rad$ for the quaternion components, is also added. An initial estimate for the DH parameters is then used to build the robot model. In addition, some bounds on the value of the DH parameters are also imposed, which are needed to use the *QP* solver.

Figure 4 shows the results by using the two currently implemented *Pinv* and *QP* solvers. For the calibration of the 3DOF arm, only the θ_i and a_i for each joint are optimized, and only the x, y Cartesian components of the tip position and the orientation about z are considered, due to the planar nature of the robot. The DH parameters and motion component selection is possible by properly choosing the entries in the matrices $\boldsymbol{W}_p$ and $\boldsymbol{W}$ used in (16) and

TABLE I: DH parameters for building the simulated robot kinematic models of: Ia) planar 3DOF robot; Ib) 6 DOF Stanford robot. The d_i, a_i are in m, α_i, θ_i in rad and the values in brackets indicate the initial guesses.

(a) DH parameters for the planar 3 DOF robot

joint	**1**	**2**	**3**
d	0	0	0
θ	0	0	0
a	1 (0.9)	0.5 (0.4)	2 (1.9)
α	0	0	0

(b) DH parameters for the 6 DOF Stanford robot

joint	**1**	**2**	**3**	**4**	**5**	**6**
d	1 (0.9)	1.5 (1.35)	0 (0)	0.5 (0.3)	0 (0)	0.1 (0.05)
θ	$-\pi/2$ ($-\pi/2$)	π (π)	0 (0)	$-\pi/2$ ($-\pi/2$)	$-\pi/2$ ($-\pi/2$)	0 (0)
a	0 (0.1)	0 (0.05)	0 (0)	0 (0.05)	0 (0.7)	0 (0)
α	$-\pi/2$ ($-\pi/2$)	$-\pi/2$ ($-\pi/2$)	0 (0)	$\pi/2$ ($\pi/2$)	$-\pi/2$ ($-\pi/2$)	0 (0)

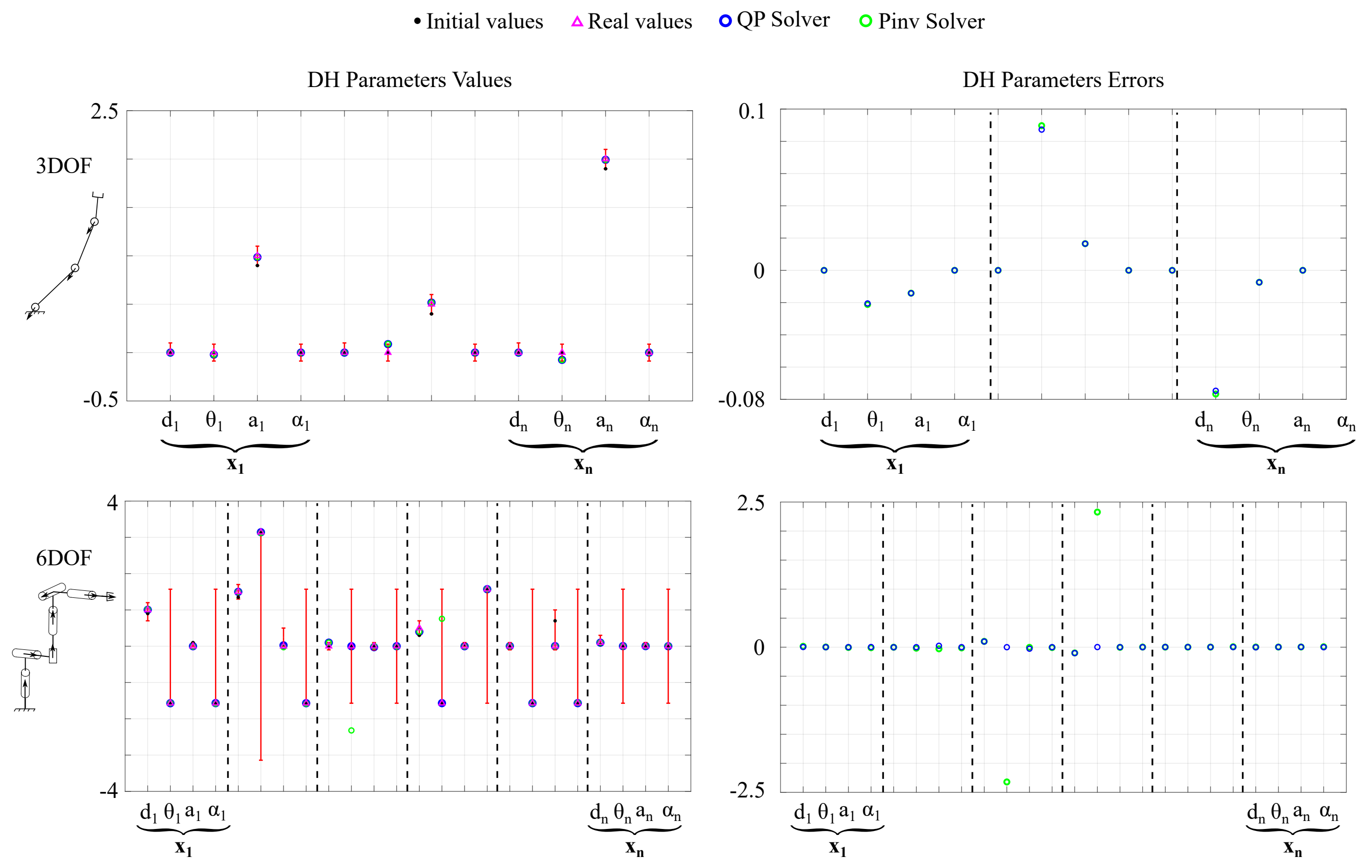


Fig. 4: Comparison of the results for the kinematic calibration of the 3DOF and the 6DOF arm using both the **Pinv** and *QP* solvers. The initial guesses are shown by black dots; the red vertical lines indicate the imposed bounds on the parameters. The errors are between the real values of the simulated DH parameters and the computed ones. The values of d_i, a_i are in m, and α_i, θ_i in rad.

(17). For the calibration of the 6 DOF Stanford manipulator, instead, all the parameters and all the motion components (both position and orientation) are chosen.

Results show that the solvers perform very similarly and well in the calibration of the 3 DOF planar robot, with small errors between the simulated DH parameters and those computed by the solvers. Some larger errors occur in the estimation of θ_2 and θ_3, with the *QP* solver yielding slightly better values. On the calibration of the 6 DOF robot, instead, adding some bounds on the values of the DH parameters leads to better performances. In fact, the errors obtained by the *QP* solver result to be smaller than those from *Pinv* solver, especially for θ_3 and θ_4. On the calibration of the 3DOF arm, both solvers converged in 11 iterations, whereas on the 6DOF QP converged in 26 iterations and $Pinv$ in 22 (Figure 6). Finally, Figure 5 shows an example of the final identification matrix $\mathbf{K}$ and it indicates that for the 3 DOF robot d_i and α_i are not identifiable. In fact, they have not been selected for the optimization. The other parameters, instead, can be identified independently. For the 6 DOF Stanford manipulator, given the current configuration and data points, there are two parameters that cannot be identified independently: a_3 and d_4 can only be identified in linear combination with other parameters.

B. Real Robot Kinematic Calibration

Kalibrot has also been used to calibrate a real KUKA LBR IIWA 14 robotic arm, starting with some incorrect initial DH parameters, corresponding to a KUKA LBR IIWA 7 arm. Similarly to the simulation case, in order to collect the data, all combinations of joints are excited with a sinusoidal motion. The tip positions are collected by using Vicon Markers (Figure 7) and Vicon tracking system [23], yet any

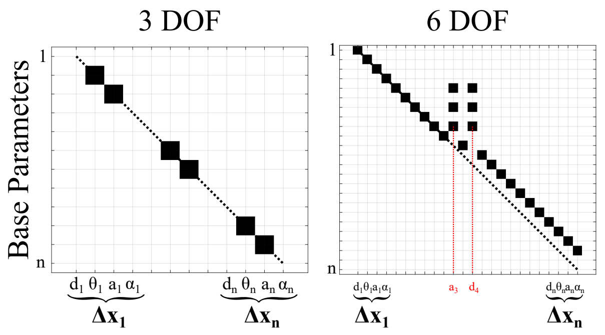


Fig. 5: Example of the final identification matrix obtained with the *Pinv* solver for the calibration of the simulated 3 DOF and 6 DOF robots showing.

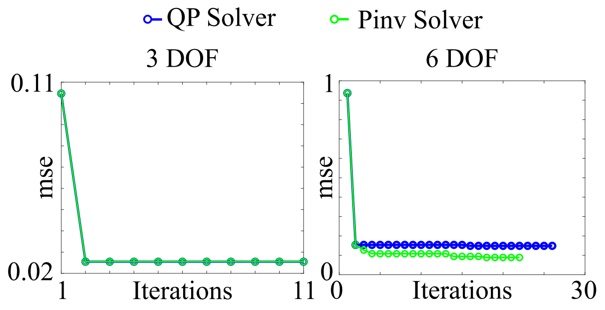


Fig. 6: Error convergence for both QP and $Pinv$ solvers. The error considers both position and orientation.

other sensors could also be used. The markers are placed on the robot's end-effector and on the robot's base, with those at the base positioned in such a way that the center of the polygon they define can geometrically locate the robot's origin. This also allows the measurement from the markers at the tip to be referred with respect to the robot's base frame. In total 63500 data points were collected and divided into a training set (80%) and a test set (20%).

Table II reports the Mean Squared Errors (MSE) on both datasets. Both QP and $Pinv$ solvers allow to have very small errors on both datasets and yield similar DH values, as shown in Figure 8a. In this test QP solver converged in 12 iterations and $Pinv$ in 6, yet QP solver manages to produce smaller offsets in the θ and α values. Figure 8b shows the Cartesian trajectory on a subset of the test dataset when using the robot model with the initial DH parameters and with the calibrated parameters from both solvers. Table III reports the initial and the calibrated DH parameters. Results show that the initial model has large errors especially in the z direction. The MSE with the initial model on the test dataset result to be $21.6mm^2$, $5.94mm^2$, and $1469mm^2$ along x, y, z respectively. Thanks to the calibration smaller errors are achieved.

TABLE II: MSE (in mm^2) from the QP and $Pinv$ solvers for the real KUKA robot arm on the training and test datasets.

	Training Set			**Test Set**		
	x	**y**	**z**	**x**	**y**	**z**
QP	1.63	0.17	0.07	1.59	0.17	0.07
Pinv	1.63	0.16	0.06	1.59	0.16	0.07

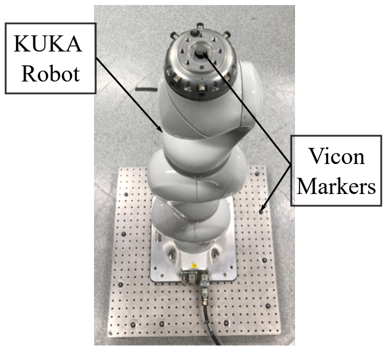


Fig. 7: Exemplary setup for data acquisition on the real KUKA arm.

TABLE III: The DH parameters on the real KUKA arm. The d, a are in m and θ, α in rad: IIIa) real and initial (in brackets); IIIb) $Pinv$ solver; IIIb) QP solver.

(a) Real and initial DH parameters

joint	**1**	**2**	**3**	**4**	**5**	**6**	**7**
d	0.36 (0.34)	0 (0)	0.42 (0.40)	0 (0)	0.40 (0.40)	0 (0)	0.126 (0.126)
θ	π (π)	π(π)	0 (0)	π (π)	0 (0)	π (π)	0 (0)
a	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
α	$\pi/2(\pi/2)$	$\pi/2(\pi/2)$	$\pi/2(\pi/2)$	$\pi/2(\pi/2)$	$\pi/2(\pi/2)$	$\pi/2(\pi/2)$	0 (0)

(b) DH parameters from $Pinv$ solver

joint	**1**	**2**	**3**	**4**	**5**	**6**	**7**
d	0.3562	0.0038	0.4239	-0.0035	0.402	-0.004	0.1269
θ	3.1395	-3.1212	0	-3.1404	-0.0021	3.1417	0.0011
a	-0.001	-0.0179	0.0099	0.0088	-0.0014	-0.0015	0
α	1.5705	1.5720	1.5653	1.5585	1.5978	1.5669	0

(c) DH parameters from QP solver

joint	**1**	**2**	**3**	**4**	**5**	**6**	**7**
d	0.3562	0.0040	0.4240	0	0.4021	0	0.1269
θ	3.1393	3.1416	0	3.1344	-0.0024	3.1408	0
a	-0.0012	0	0	0.0028	0	0	0
α	1.5706	1.5725	1.5651	1.5650	1.5881	1.5887	0

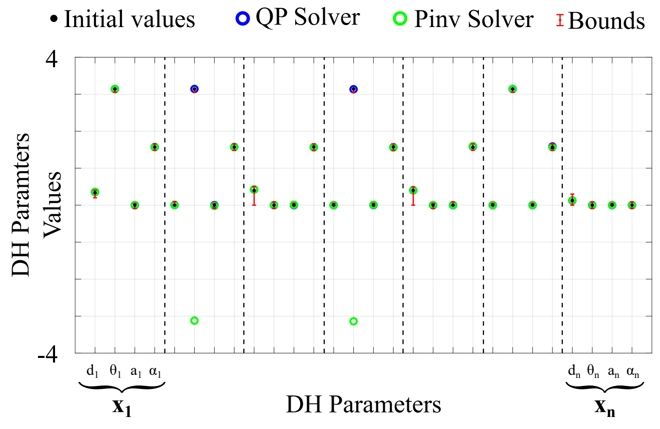
V. Conclusions

In conclusion, this paper presented Kalibrot, an open-source Matlab package for solving the problem of robot kinematic calibration. Kalibrot is an easy-to-use package whose goal is simplifying robot kinematic calibration procedure, especially for inexperienced users. The architecture of Kalibrot allows for:

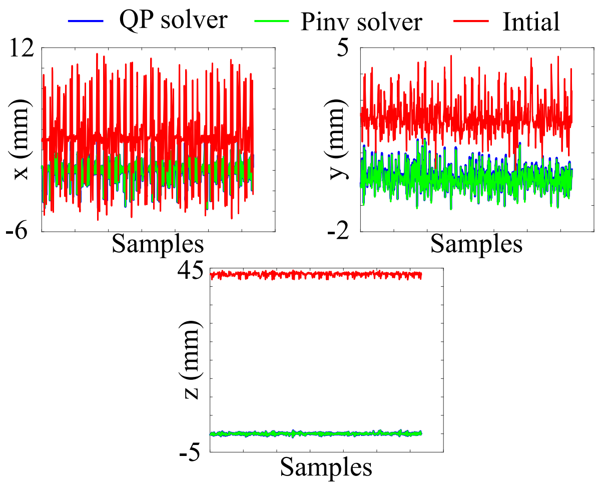
- simple definition of a robot model by using the $RobotKinematics$ class;
- easy calibration setup with the possibility to choose between different solvers, select what parameters to optimize for, and what Cartesian component to consider;
- visualization and analysis of the results by means of the visualization layer which plots the calibrated DH parameters and the calibrated kinematic structure;
- additional assessment of the calibration results by returning the the identification matrix and observability measures to analyse what parameters can be identified and the accuracy of the calibration.

In this manuscript, the capabilities of Kalibrot have been shown by performing calibrations of different robot types, both in simulation and in a real world scenario.

At the current stage, Kalibrot deals only with optimization of the DH parameters and provides two solvers. However



(a) The DH parameters



(b) Comparison of the tip position errors on a subset of the test dataset

Fig. 8: Results for the calibration of the real KUKA robot arm: 8a) comparison of the calibrated DH parameters with $Pinv$ and QP solvers.The d_i, a_i are in m, α_i, θ_i in rad; 8b) comparison of the Cartesian trajectory between the collected data, the initial model, and the calibrated models.

all components in Kalibrot are written in a flexible and extensible way, which allows user-specific definition. Different robotic structures can be implemented by editing the $RobotKinematics$ class and additional solvers can be included. We would like to encourage the robotic research community to take advantage of this research-friendly software and contribute to improving it and the availability of calibration datasets .

ACKNOWLEDGMENT

The authors would like to thank Xiao Guo, from Hamlyn Centre, Imperial College London, for his help in collecting the data with Vicon tracking system.

REFERENCES

[1] J. Cheong, W. Chung, and Y. Youm, "Inverse Kinematics of Multilink Flexible Robots for High-Speed Applications," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 269–282, 4 2004.

[2] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum Robots for Medical Applications: A Survey," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1261–1280, 2015.

[3] W. Bai, Q. Cao, P. Wang, P. Chen, C. Leng, and T. Pan, "Modular design of a teleoperated robotic control system for laparoscopic minimally invasive surgery based on ROS & RT-Middleware," *Industrial Robot*, vol. 44, no. 5, pp. 596–608, 2017.

[4] W. Bai, Q. Cao, C. Leng, Y. Cao, M. G. Fujie, and T. Pan, "A novel optimal coordinated control strategy for the updated robot system for single port surgery," *International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 13, no. 3, 9 2017.

[5] B. Chen, Q. Cao, and W. Bai, "A design of surgical robotic system based on 6-DOF parallel mechanism," in *International Conference on Biological Information and Biomedical Engineering (BIBE 2018)*, Shangai, 2018.

[6] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics*, ser. Advanced Textbooks in Control and Signal Processing. London: Springer London, 2009.

[7] Z. Roth, B. Mooring, and B. Ravani, "An overview of robot calibration," *IEEE Journal on Robotics and Automation*, vol. 3, no. 5, pp. 377–385, 10 1987.

[8] A. Joubair, A. Nubiola, and I. Bonev, "Calibration Efficiency Analysis Based on Five Observability Indices and Two Calibration Models for a Six-Axis Industrial Robot," *SAE International Journal of Aerospace*, vol. 6, no. 1, pp. 2013–01, 9 2013.

[9] A. Joubair, M. Slamani, and I. A. Bonev, "A novel XY-Theta precision table and a geometric procedure for its kinematic calibration," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 1, pp. 57–65, 2 2012.

[10] M. Švaco, B. Šekoranja, F. Šuligoj, and B. Jerbić, "Calibration of an Industrial Robot Using a Stereo Vision System," *Procedia Engineering*, vol. 69, pp. 459–463, 1 2014.

[11] A. Joubair and I. A. Bonev, "Kinematic calibration of a six-axis serial robot using distance and sphere constraints," *The International Journal of Advanced Manufacturing Technology*, vol. 77, no. 1-4, pp. 515–523, 3 2015.

[12] P. T. Katsiaris, G. Adams, S. Pollard, and S. J. Simske, "A kinematic calibration technique for robotic manipulators with multiple Degrees of Freedom," in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 7 2017, pp. 358–363.

[13] G. Du, H. Shao, Y. Chen, P. Zhang, and X. Liu, "An online method for serial robot self-calibration with CMAC and UKF," *Robotics and Computer-Integrated Manufacturing*, vol. 42, pp. 39–48, 12 2016.

[14] Ruibo He, Yingjun Zhao, Shunian Yang, and Shuzi Yang, "Kinematic-Parameter Identification for Serial-Robot Calibration Based on POE Formula," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 411–423, 6 2010.

[15] J. M. Hollerbach and C. W. Wampler, "The Calibration Index and Taxonomy for Robot Kinematic Calibration Methods," *The International Journal of Robotics Research*, vol. 15, no. 6, pp. 573–591, 12 1996.

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

[17] J. J. Moré, "The Levenberg-Marquardt algorithm: Implementation and theory." Springer, Berlin, Heidelberg, 1978, pp. 105–116.

[18] M. Gautier, "Numerical calculation of the base inertial parameters of robots." Publ by IEEE, 1990, pp. 1020–1025.

[19] C. H. Menq, J. H. Borm, and J. Z. Lai, "Identification and observability measure of a basis set of error parameters in robot calibration," *Journal of Mechanical Design, Transactions of the ASME*, vol. 111, no. 4, pp. 513–518, 12 1989.

[20] M. Gautier and W. Khalil, "Exciting Trajectories for Robot Inertial Parameters Identification," *IFAC Proceedings Volumes*, vol. 25, no. 15, pp. 585–590, 7 1992.

[21] C. Mummolo, F. Cursi, and J. Kim, "Balanced and falling states for biped systems: Applications to robotic versus human walking stability," in *IEEE-RAS International Conference on Humanoid Robots*, 2016.

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

[23] P. Merriaux, Y. Dupuis, R. Boutteau, P. Vasseur, and X. Savatier, "A study of vicon system positioning performance," *Sensors (Switzerland)*, vol. 17, no. 7, 7 2017.