Automatic Denavit-Hartenberg Parameter Identification for Serial Manipulators

Carlos Faria
DTx-Colab, Centre Algorithmi and
Department of Industrial Electronics
University of Minho
Guimarães, Portugal
cfaria@dei.uminho.pt

João L. Vilaça
2Ai
Polytechnic Institute of Cávado and Ave
Barcelos, Portugal
jvilaca@ipca.pt

Sérgio Monteiro
Centre Algorithmi and
Department of Industrial Electronics
University of Minho
Guimarães, Portugal
sergio@dei.uminho.pt

Wolfram Erlhagen
Centre of Mathematics and
Department of Mathematics
University of Minho
Guimarães, Portugal
wolfram.erlhagen@math.uminho.pt

Estela Bicho
Centre Algorithmi and
Department of Industrial Electronics
University of Minho
Guimarães, Portugal
estela.bicho@dei.uminho.pt

***Abstract*—An automatic algorithm to identify Standard Denavit-Hartenberg parameters of serial manipulators is proposed. The method is based on geometric operations and dual vector algebra to process and determine the relative transformation matrices, from which it is computed the Standard Denavit-Hartenberg (DH) parameters (a_i, α_i, d_i, θ_i). The algorithm was tested in several serial robotic manipulators with varying kinematic structures and joint types: the KUKA LBR iiwa R800, the Rethink Robotics Sawyer, the ABB IRB 140, the Universal Robots UR3, the KINOVA MICO, and the Omron Cobra 650. For all these robotic manipulators, the proposed algorithm was capable of correctly identifying a set of DH parameters. The algorithm source code as well as the test scenarios are publicly available.**

***Index Terms*—Kinematic identification, Denavit-Hartenberg parameters**

I. Introduction

Kinematic identification refers to the determination of a minimal number of parameters that completely describe the position and orientation of a manipulator's structure as a function of its joint positions. Many models have been proposed to characterize a kinematic structure: the Standard and Modified Denavit-Hartenberg (DH) convention [1], the Hayati model [2], the Stone and Sanderson's S-model [3] and recent models based on Product of Exponentials (POE) [4], [5].

The Denavit-Hartenberg model is still the most used convention to represent the robot's kinematic structure. It provides a guaranteed minimal representation, an intuitive method to determine its parameters and most importantly, it works on straight-forward linear algebra whose matrices are computationally fast to solve.

This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2019, the FCT scholarship grant: SFRH/BD/86499/2012 and the DTx-Colab.

Independent of the convention used, there is a consistent problem with robots, particularly serial structures that causes repeatable but inaccurate movements. This problem derives from manufacturing and assembly tolerances, wear and tear, or permanent bending due to fatigue. The listed error sources are reflected on the real kinematic model parameters, and on the gap to the nominal parameter values. Kinematic errors are especially impactful in serial manipulators where the parameter deviations propagate through the kinematic chain. Industrial robot calibration methods, particularly the ones based on kinematic parameters, are compartmentalized in four steps, i) modelling, ii) measurement, iii) identification, and compensation [6], [7]. The proposed algorithm is primarily related to the parameter identification step as a sub-type of planar calibration methods [8], [9].

Another common problem to users that need to model kinematic structures of robotic manipulators relates to the inaccessibility of the model parameters, which are usually handled internally by the controller. Even if the robot is properly calibrated, the user has no access to the kinematic parameters other than the nominal values in the documentation.

In this paper, we propose an algorithm for DH parameter identification based on geometry and dual vector algebra for any type of serial robot. The algorithm splits into two parts, the first called "feature identification". In this part, the robot's end-effector position is acquired after sequential movements in each joint. The acquired set of points are processed to determine the motion axis of each joint, an idea originally explored by Stone [3] to determine the S-model parameters. The second part, "parameter extraction", applies dual vector algebra to calculate the intermediate coordinate frames between consecutive joints, and then to extrapolate the Standard DH parameters. The dual vector algebra concept was applied

978-1-7281-4878-6/19/$31.00 ©2019 IEEE

by Ketchel and Larochelle [10] to detect collisions between robotic links modeled after cylindrical bodies.

To the best of our knowledge, the proposed algorithm is the first capable of correctly identifying a set of DH parameters for a wide range of differentiated robots, representing various kinematic typologies. The method was tested against serial robotic manipulators with: revolute and prismatic joints, intrinsically redundant and non-redundant structures, joints with aligned motion axis, shoulder and wrist offsets, as well as with curved wrists. It corrects and extends previous work by [11], [12], which fails to deliver a correct set of DH parameters for manipulators other than the industrial serial 6-DoF type. The source code of the algorithm here proposed is available at https://github.com/neuebot/Kinematic-Calibration.

The proposed method was tested with several robot models with variable structures and joint dispositions including a KUKA LBR iiwa R800, a Rethink Robotics Sawyer, an ABB IRB 140, a Universal Robots UR3, a KINOVA MICO and a SCARA-type Cobra 650 by Omron. A correct set of DH parameters was identified for all.

The paper is organized as follows. Section II describes the process of identifying the motion axis for a generic prismatic or revolute joint. In section III it is explained how to determine the relative transformation frames between joints from which the DH parameters are extracted. Section IV presents the results of the proposed method for different types of serial manipulators.

II. Feature Identification

The objective of this section is to determine, for each joint of the manipulator, its motion axis: a) for a revolute joint - the plane and center of rotation; b) for a prismatic joint - the sliding vector and a contained point. These values constitute the input to the proposed algorithm.

To identify the motion axis of each joint, we move each robot joint separately and acquire the position of a point ($\mathbf{p} = (x, y, z)^T$) at the robot's end-effector relative to the robot's base reference frame. Either $\mathbf{p}$ is determined directly from the robot controller, which allows for the extraction of the controller's intrinsic kinematic parameters, or $\mathbf{p}$ is determined using an external tracking sensor to achieve kinematic calibration. For each joint, from the base to the end-effector, a movement in joint space is executed, which translates to a circular or linear trajectory of the tracked point ($\mathbf{p}$) in Cartesian space depending on the type of the joint.

For consistent and reliable results, our method imposes the following requirements:

1) the reference frame attached to the first joint should be aligned with the base reference frame;
2) the joint motion must be in the positive direction during the motion acquisition;
3) the acquired position measurement step should be 3 to 4 orders of magnitude less than the nominal distance of the generated trajectory;
4) the acquired point must be contained in the plane defined by the normal to the last joint axis that contains the end-effector position, see Fig. 1.

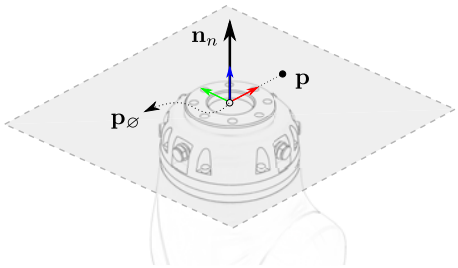

Fig. 1: Example of acquired point ($\mathbf{p}$) contained in the plane defined by the last joint axis ($\mathbf{n}_n$) and the position of the flange. The acquired point must not be contained in the line that passes through the last joint axis, i.e. $\mathbf{p} \neq \mathbf{p}_\varnothing$.

The last condition assures that the tracked point trajectory generated from the last joint is not reduced to a single point. In addition, for the method here proposed, we recommend that:

1) each joint travels more than a third of its range;
2) each joint travels at identical and constant velocities - to guarantee a similar number of samples and uniform distribution of end-effector positions.

After acquiring the generated trajectories of $\mathbf{p}$ for each isolated joint, one should end up with n trajectories (n equates to the number of joints), each with m number of points corresponding to the tracked point measurements, $\mathbf{P} = \{\mathbf{p}_j \in \mathbb{R}^3, j \in \{1, \ldots, m\}\}$.

Since the process is iterative to each joint, we explain in the following section how it applies to a generic prismatic and a revolute joint.

A. Prismatic Joints

A prismatic joint motion will generate a linear trajectory in workspace. The motion axis vector can be determined by finding the best fitting line to the set of trajectory points. Line and plane fitting are problems commonly addressed with Orthogonal Distance Regression. The goal is to minimize the distances between the set of points and the geometric element. Let the position of the best fitting line be represented by a point $\mathbf{c}$ belonging to the line, and let the vector $\mathbf{n}$ be the direction vector. The orthogonal distance (d_j) between each point ($\mathbf{p}_j$) and the line is,

$$d_j = \|(\mathbf{p}_j - \mathbf{c}) - ((\mathbf{p}_j - \mathbf{c}) \cdot \mathbf{n})\,\mathbf{n}\|, \tag{1}$$

assuming,

$$\mathbf{c} = \frac{1}{m}\sum_{j=1}^{m} \mathbf{p}_j. \tag{2}$$

and without the loss of generality $\mathbf{n}$ to be a unit vector (i.e. $\|\mathbf{n}\| = 1$). For clarity, consider the vector norm $\|\bullet\|$ to be the Euclidean norm.

The best fitting line minimizes the square sum of orthogonal distances between the line and the points,

$$\min_{\mathbf{n}} \sum_{j=1}^{m} {d_j}^2$$

equivalent to,

$$\min_{\mathbf{n}} \left[\sum_{j=1}^{m} \|\mathbf{p}_j - \mathbf{c}\|^2 - \sum_{j=1}^{m} ((\mathbf{p}_j - \mathbf{c}) \cdot \mathbf{n})^2 \right]$$

that can be written as,

$$\min_{\mathbf{n}} \left(\|\mathbf{M}\|_F - \|\mathbf{M}\mathbf{n}\|^2 \right),$$

where $\mathbf{M}$ is the matrix of $3 \times m$ mean-centered points, i.e.,

$$\mathbf{M} = \begin{bmatrix} \mathbf{p}_1 - \mathbf{c} \\ \vdots \\ \mathbf{p}_m - \mathbf{c} \end{bmatrix} \tag{3}$$

and $\| \bullet \|_F$ is the Frobenius norm. Thus, the problem translates to finding,

$$\mathbf{n} := \arg\max_{\substack{\mathbf{n} \in \mathbb{R}^3 \\ \|\mathbf{n}\|=1}} \|\mathbf{M}\mathbf{n}\|^2. \tag{4}$$

One efficient approach to deal with this problem relies on Singular Value Decomposition (SVD) to determine the dominant direction of data. Factorizing $\mathbf{M}$ into the product of three matrices $\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices whose columns are orthonormal and Σ is a diagonal matrix with positive and real singular values listed in decreasing order ($\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$). It follows that,

$$\|\mathbf{M}\mathbf{n}\|^2 = \|\mathbf{U}\Sigma\mathbf{V}^T\mathbf{n}\|^2 \tag{5}$$

and given that $\mathbf{U}$ is an unitary matrix, the following is also true,

$$\|\mathbf{M}\mathbf{n}\|^2 = \|\Sigma\mathbf{V}^T\mathbf{n}\|^2.$$

Considering $\mathbf{h} = \mathbf{V}^T\mathbf{n}$, the function (4) is equivalent to,

$$\mathbf{n} := \arg\max_{\substack{\mathbf{n} \in \mathbb{R}^3 \\ \|\mathbf{n}\|=1}} \left[(\sigma_1 h_1)^2 + (\sigma_2 h_2)^2 + (\sigma_3 h_3)^2 \right]. \tag{6}$$

Provided the decreasing order of singular values, the function is maximized for $\mathbf{h}_{\max} = [1, 0, 0]^T$, which follows that,

$$\mathbf{n} := \mathbf{V}\mathbf{h}_{\max}. \tag{7}$$

The tail-arrow direction of $\mathbf{n}$ is also important because it relates to the direction of the axis. As one of the imposed requirements for the calibration was to record a linear joint motion in the positive direction, one can guarantee the correct tail-arrow direction of $\mathbf{n}$ by,

$$\mathbf{n} = \begin{cases} \mathbf{n}, & \text{if } \mathbf{n} \cdot (\mathbf{p}_m \times \mathbf{p}_1) \geq 0 \\ -\mathbf{n}, & \text{if } \mathbf{n} \cdot (\mathbf{p}_m \times \mathbf{p}_1) < 0 \end{cases}. \tag{8}$$

B. Revolute Joints

One logical approach to calculate the plane and center of rotation is to determine the best fitting tri-dimensional circle to the trajectory, from which it is straightforward to extrapolate the plane and center of rotation. The first step consists in determining the best fitting plane to the set of points $\mathbf{P}$. To keep a consistent notation, let this plane be represented by its normal vector $\mathbf{n}$, and a plane contained point $\mathbf{c}$ determined from the centroid of the set of points, similar to the previous subsection. The orthogonal distances between the plane and the set of points are,

$$d_j = (\mathbf{p}_j - \mathbf{c}) \cdot \mathbf{n}. \tag{9}$$

The problem of finding the best fitting plane is similar to the one described in subsection II-A,

$$\mathbf{n} := \arg\min_{\substack{\mathbf{n} \in \mathbb{R}^3 \\ \|\mathbf{n}\|=1}} \|\mathbf{M}\mathbf{n}\|^2. \tag{10}$$

which simplifies to,

$$\mathbf{n} := \arg\min_{\substack{\mathbf{n} \in \mathbb{R}^3 \\ \|\mathbf{n}\|=1}} \left[(\sigma_1 h_1)^2 + (\sigma_2 h_2)^2 + (\sigma_3 h_3)^2 \right]. \tag{11}$$

that is minimized for $\mathbf{h}_{\min} = [0, 0, 1]^T$,

$$\mathbf{n} := \mathbf{V}\mathbf{h}_{\min}. \tag{12}$$

Again and in a similar fashion to the previous subsection, the tail-arrow direction of the plane vector can be correctly determined using equation (8).

Now, since the best fitting plane for the points $\mathbf{P}$ has been determined, we can determine the circle that best fits these points in two dimensions, i.e projected into the plane defined by $\mathbf{n}$. First, the transformation of $\mathbf{P}$ to the best fitting plane coordinates, $\mathbf{n}$, is achieved with the *Rodrigues' rotation formula*,

$$\mathbf{p}_{\mathbf{n},j} = \mathbf{p}_j \cos\rho + (\mathbf{a} \times \mathbf{p}_j) \sin\rho + \mathbf{a}(\mathbf{a} \cdot \mathbf{p}_j)(1 - \cos\rho), \tag{13}$$

which rotates $\mathbf{P}$ around an axis $\mathbf{a}$ of an angle ρ,

$$\mathbf{a} = \mathbf{n} \times [0, 0, 1]^T \tag{14}$$

$$\rho = \operatorname{atan}_2 \left(\|\mathbf{n} \times [0, 0, 1]^T\|, \mathbf{n} \cdot [0, 0, 1]^T \right). \tag{15}$$

The projection of these points into the plane coordinates, XY-plane, is then simply accomplished by using the x and y coordinates.

With the projected points into the XY plane - $\mathbf{P_n} = \left\{ \mathbf{p}_{\mathbf{n},j} \in \mathbb{R}^3, j \in \{1, \ldots, m\} \right\}$, it is straightforward to determine the best 2D fitting circle. Consider a circle of radius r and center (x_c, y_c), which is represented by,

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \tag{16}$$

simplified as a function of x and y, one ends up with,

$$(2x_c)x + (2y_c)y + (-x_c^2 - y_c^2 + r^2) = x^2 + y^2. \tag{17}$$

We can now transform this into a system of linear first degree equations using the projected points $\mathbf{p}_{\mathbf{n},j} = (x_j, y_j, 0)^T$ to

determine the circle center coordinates and radius. Converting (17) to matrix notation ($\mathbf{Ax} = \mathbf{b}$),

$$\mathbf{A} = \begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_m & y_m & 1 \end{bmatrix}, \ \mathbf{x} = \begin{bmatrix} 2x_c \\ 2y_c \\ -x_c^2 - y_c^2 + r^2 \end{bmatrix},$$

$$\mathbf{b} = \begin{bmatrix} x_1^2 + y_1^2 \\ \vdots \\ x_m^2 + y_m^2 \end{bmatrix}.$$

The system is solved as a function of $\mathbf{x}$ through a least-squares fitting approach. The goal is to minimize the square sum of residual errors $\|\mathbf{b} - \mathbf{Ax}\|^2$. It is now straightforward to find the center and radius of the circle once $\mathbf{x}$ is determined.

Having determined the best fitting circle in the plane defined by $\mathbf{n}$, it can be directly transformed to tri-dimensional coordinates by applying the inverse transformation to (13), i.e. inverting the axis and angle of rotation.

III. Parameter Extraction

With the vector $\mathbf{n}$ and the point $\mathbf{c}$ determined relative to the robot base reference frame and for each joint motion, one can extrapolate the Denavit-Hartenberg parameters using a geometric approach that consists of two steps:

1) determine the relative coordinate frames that relate each actuated joint;
2) identify the DH parameters from the set of frames.

The relative coordinate frames (18) can be partially constructed from the information gathered thus far.

$${}^{i}\mathbf{T}_{i+1} = \begin{bmatrix} \hat{\mathbf{x}}_{i+1} & \hat{\mathbf{y}}_{i+1} & \hat{\mathbf{z}}_{i+1} & \mathbf{p}_{i+1} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{18}$$

where

$$\hat{\mathbf{z}}_{i+1} = \mathbf{n}_{i+1}$$
$$\hat{\mathbf{y}}_{i+1} = \hat{\mathbf{z}} \times \hat{\mathbf{x}}$$

Per definition, the z-axis should equate to the direction of the joint axis ($\mathbf{n}_i$). Similarly, the x-axis can be determined from the common normal to both direction axes, although this vector might not be uniquely determined. From the right hand rule the y-axis is directly determined once the z- and the x-axes are known. On the other hand, determining the position vector of each coordinate frame requires some Dual Vector geometrical operations to calculate the intersection/closest points between the axis lines.

Each pair of parameters $(\mathbf{n}_i, \mathbf{c}_i)$ forms the base to represent a line in tri-dimensional space. Alternatively, the same information can be formulated in Plücker coordinates (S) using the notation of the moment vector ($\mathbf{k}$),

$$S = \begin{bmatrix} \mathbf{n} \\ \mathbf{c} \times \mathbf{n} \end{bmatrix} = \begin{bmatrix} \mathbf{n} \\ \mathbf{k} \end{bmatrix}. \tag{19}$$

Analogous to complex numbers, dual number notation can be expressed by a sum of two parts: the primary component or real part ($\mathbf{n}$) and the dual component or dual part ($\mathbf{k}$) [13]:

$$S = \mathbf{n} + \varepsilon\ \mathbf{k} \tag{20}$$

where $\varepsilon \neq 0$ and $\varepsilon^2 = 0$. This representation is especially useful to study the relationship of two straight lines in tri-dimensional space, Fig. 2. Ketchel and Larochelle [10] proposed an algorithm to classify this relationship based on dual vector representation for the purpose of collision detection. Using dual vector algebra to determine the dot and cross

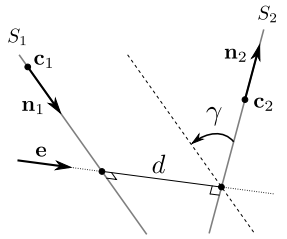

Fig. 2: The relationship between two lines in space can be parameterized by the shortest distance between lines d (dual part), and the projected angle between the lines γ (real part) [10]. The common normal is, $\hat{\mathbf{e}} = \|\mathbf{n}_1 \times \mathbf{n}_2\|$. The distance between lines is, $d = (\mathbf{c}_2 - \mathbf{c}_1) \cdot \mathbf{e}$. The angle between lines can be determined from, $\gamma = \operatorname{atan2}(\|\mathbf{n}_1 \times \mathbf{n}_2\|, \mathbf{n}_1 \cdot \mathbf{n}_2)$.

products of S_1 and S_2 one can determine the distance (d) and angle (γ) that relate them,

$$\begin{aligned} \hat{S}_1 \cdot \hat{S}_2 &= (\mathbf{n}_1, \mathbf{k}_1) \cdot (\mathbf{n}_2, \mathbf{k}_2) \\ &= (\mathbf{n}_1 \cdot \mathbf{n}_2, \mathbf{n}_1 \cdot \mathbf{k}_2 + \mathbf{k}_1 \cdot \mathbf{n}_2) \\ &= \mathbf{n}_1 \cdot \mathbf{n}_2 + \varepsilon(\mathbf{n}_1 \cdot \mathbf{k}_2 + \mathbf{k}_1 \cdot \mathbf{n}_2) \\ &= \cos\gamma - \varepsilon\ d \sin\gamma \end{aligned}$$

$$\begin{aligned} \hat{S}_1 \times \hat{S}_2 &= (\mathbf{n}_1, \mathbf{k}_1) \times (\mathbf{n}_2, \mathbf{k}_2) \\ &= (\mathbf{n}_1 \times \mathbf{n}_2, \mathbf{n}_1 \times \mathbf{k}_2 + \mathbf{k}_1 \times \mathbf{n}_2) \\ &= \mathbf{n}_1 \times \mathbf{n}_2 + \varepsilon(\mathbf{n}_1 \times \mathbf{k}_2 + \mathbf{k}_1 \times \mathbf{n}_2) \\ &= (\sin\gamma - \varepsilon\ d \cos\gamma)\ \hat{\mathbf{e}}. \end{aligned}$$

S_1 and S_2 are either: **intersecting**, **identical**, **parallel**, or **skewed**; best explained in the flowchart Fig. 3.

A. Intersecting lines

The simplest case involves **intersecting lines**, as it logically follows that the position vector should be located at the intersection point,

$$\mathbf{p}_{\text{int}} = \begin{cases} \dfrac{\mathbf{k}_1 \times \mathbf{k}_2}{\mathbf{n}_2 \cdot \mathbf{k}_1}, & \text{if } \mathbf{n}_1 \cdot \mathbf{k}_2 - \mathbf{n}_2 \cdot \mathbf{k}_1 \neq 0 \\ \dfrac{\mathbf{k}_2 \times \mathbf{k}_1}{\mathbf{n}_1 \cdot \mathbf{k}_2}, & \text{if } \mathbf{n}_1 \cdot \mathbf{k}_2 - \mathbf{n}_2 \cdot \mathbf{k}_1 = 0 \end{cases}. \tag{21}$$

While (21) returns a solution for most cases, if any of the lines passes through the origin of the reference system, its moment vector becomes null ($\mathbf{k} = 0$) making it impossible to determine $\mathbf{p}_{it}$. Rajeevlochana et al. [12] proposed a solution for this special case, which involves computing an auxiliary intersection point at an auxiliary coordinate frame and then applying the inverse transformation to obtain the real intersection point. The auxiliary coordinate frame (${}^{1}\mathbf{T}_{1'}$) is found by applying a pure

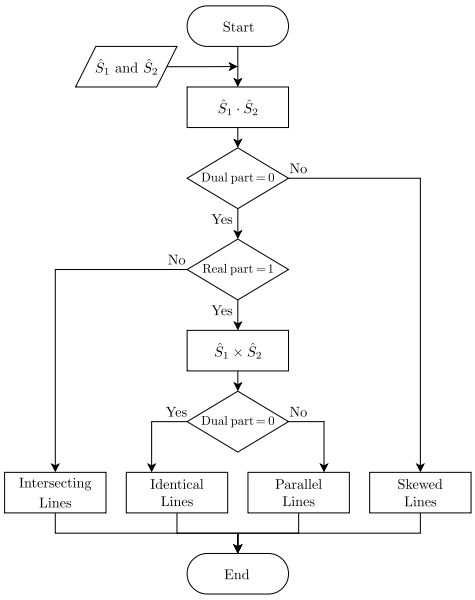

Fig. 3: The line relationship categorization process.

translation of a unit distance from the reference frame along the common normal of the vectors $\mathbf{n}_1$ and $\mathbf{n}_2$,

$$ {}^{1}\mathbf{T}_{1'} = \begin{bmatrix} & \mathbf{I}_3 & & \|\mathbf{n}_1 \times \mathbf{n}_2\| \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{22} $$

The computation of the auxiliary intersection point in the auxiliary reference frame is exactly the same as for the normal case (23), the only difference being the translation of the auxiliary center of rotations coordinates ($\mathbf{c}_{1'}$ and $\mathbf{c}_{2'}$) and consequently the auxiliary moment vectors ($\mathbf{k}_{1'}$ and $\mathbf{k}_{2'}$). From the auxiliary system, the actual intersection point is determined by translating the auxiliary intersection point back to the original reference system, as given by:

$$ \mathbf{p}_{\text{int}} = \begin{cases} ({}^{1}\mathbf{T}_{1'})^{-1} \ \dfrac{\mathbf{k}_{1'} \times \mathbf{k}_{2'}}{\mathbf{n}_2 \cdot \mathbf{k}_{1'}}, & \text{if } \mathbf{n}_1 \cdot \mathbf{k}_{2'} - \mathbf{n}_2 \cdot \mathbf{k}_{1'} \neq 0 \\ ({}^{1}\mathbf{T}_{1'})^{-1} \ \dfrac{\mathbf{k}_{2'} \times \mathbf{k}_{1'}}{\mathbf{n}_1 \cdot \mathbf{k}_{2'}}, & \text{if } \mathbf{n}_1 \cdot \mathbf{k}_{2'} - \mathbf{n}_2 \cdot \mathbf{k}_{1'} = 0 \end{cases}, \tag{23} $$

where

$$ \begin{aligned} \mathbf{k}_{1'} &= \mathbf{k}_1 + (\|\mathbf{n}_1 \times \mathbf{n}_2\| \times \mathbf{n}_1) \\ \mathbf{k}_{2'} &= \mathbf{k}_2 + (\|\mathbf{n}_1 \times \mathbf{n}_2\| \times \mathbf{n}_2) \end{aligned} $$

The relative coordinate frame (18) is now fully defined for intersecting lines:

$$ \begin{aligned} \mathbf{p}_{i+1} &= \mathbf{p}_{\text{int}} \\ \hat{\mathbf{x}}_{i+1} &= \|\mathbf{n}_i \times \mathbf{n}_{i+1}\| \end{aligned} $$

B. Identical lines

Identical lines are perhaps the most uncommon of all cases (see the SCARA robot "Cobra 650" in section IV). No intersection point or common normal are uniquely determined and thus it is up to the algorithm to decide both. The relative coordinate frame (18) may be defined for identical lines according to:

$$ \begin{aligned} \mathbf{p}_{i+1} &= \mathbf{p}_i \\ \hat{\mathbf{x}}_{i+1} &= \mathbf{n}_{1'} \end{aligned} $$

C. Parallel lines

If the lines are **parallel**, there is no intersecting point and there is no unique common normal. In this case, the algorithm imposes a point through which the common normal should pass through, for example, the center of rotation of the first joint ($\mathbf{c}_1$) in the studied pair. Using Dual Vector algebra, a line ($S_{1'}$) can be drawn from $\mathbf{c}_1$ to the closest point along S_2,

$$ S_{1'} = \begin{bmatrix} \mathbf{n}_{1'} \\ \mathbf{c}_1 \times \mathbf{n}_{1'} \end{bmatrix} \tag{24} $$

where

$$ \mathbf{n}_{1'} = \|\mathbf{n}_1 \times ((\mathbf{c}_2 - \mathbf{c}_1) \times \mathbf{n}_1)\| \tag{25} $$

It is now possible to determine the intersection point between $S_{1'}$ and S_2 with the method described for intersecting lines. Note that, if both lines are parallel, the common normal vector (specified as $\hat{\mathbf{e}}$ in Fig. 2) is calculated as in (25). The relative coordinate frame (18) is now fully defined for parallel lines:

$$ \begin{aligned} \mathbf{p}_{i+1} &= \mathbf{p}_{\text{int}} \\ \hat{\mathbf{x}}_{i+1} &= \mathbf{n}_{1'} \end{aligned} $$

D. Skewed lines

Finally, if the lines are skewed we have a two-part solution representing the closest points along the lines $S1$ and $S2$ to one another,

$$ \mathbf{p}_{\text{int},1} = \left(\frac{\mathbf{k}_2 \cdot \mathbf{e} - \cos\gamma \ \mathbf{k}_1 \cdot \mathbf{e}}{\sin\gamma} \right) \mathbf{n}_1 + \mathbf{n}_1 \times \mathbf{k}_1 \tag{26} $$

$$ \mathbf{p}_{\text{int},2} = \left(\frac{-\mathbf{k}_1 \cdot \mathbf{e} + \cos\gamma \ \mathbf{k}_2 \cdot \mathbf{e}}{\sin\gamma} \right) \mathbf{n}_2 + \mathbf{n}_2 \times \mathbf{k}_2. \tag{27} $$

The skewed lines case is specially relevant for robots that have offsets. Both points are used to compute DH parameters. The common normal in these cases is determined from the vector that passes through both points.

$$ \begin{aligned} \mathbf{p}_{i+1} &= \mathbf{p}_{\text{int},2} \\ \hat{\mathbf{x}}_{i+1} &= \mathbf{n}_i \times \mathbf{n}_{i+1} \end{aligned} $$

Due to the fact that two solutions are possible for the common normal, accounting for the inverted vector, two conventions are suggested.

Whenever the common normal is aligned with the previous frame common normal, $\|\hat{\mathbf{x}}_{i+1} \cdot \hat{\mathbf{x}}_i\| = 1$, then the direction of the current common normal should match the previous, $\hat{\mathbf{x}}_{i+1} = \hat{\mathbf{x}}_i$. Else, when considering two inverted common normals with a non-zero y-component, $\|\hat{\mathbf{x}}_{i+1} \cdot [0 \ 1 \ 0]\| > 0$, we select the solution with a positive y-component.

E. Extracting DH parameters

Knowing $\mathbf{p}_{i+1}$ and $\hat{\mathbf{x}}_{i+1}$, we completely define the relative coordinate frame between actuated joints independent from the robot structure. To avoid ambiguities, the robot base frame is purposely identical to the system reference frame, implying that its origin is coincident with the origin of the first relative coordinate frame (joint 0), and its z-axis aligned with the first joint rotation axis. With the first relative coordinate frame assigned, the process of finding the remaining DH parameters is iterative for each $i \in \{0, \ldots, n-1\}$. Note that the point $\mathbf{p}_n$ (position of the last joint) as well as $\hat{\mathbf{z}}_n$ are required. This problem is abbreviated by providing an end-effector position ($\mathbf{p}_n$) as input, and assuming $\hat{\mathbf{z}}_n$ to be parallel to the last joint rotation axis ($\hat{\mathbf{z}}_{n-1}$). If not, the final frame (${}^{n-1}\mathbf{T}_n$) is also required as input.

After calculating the relative coordinate frames from the base to the robot's end-effector, the four DH parameters used to describe each transformation are:

1) d_i, offset from the previous z-axis to the common normal,
$$d_i = (\mathbf{p}_{i+1} - \mathbf{p}_i) \cdot \hat{\mathbf{z}}_i \tag{28}$$
2) θ_i, angle between the previous to the current x-axis around the previous z-axis,
$$\theta_i = \operatorname{atan2}\left((\hat{\mathbf{x}}_i \times \hat{\mathbf{x}}_{i+1}) \cdot \hat{\mathbf{z}}_i, \hat{\mathbf{x}}_i \cdot \hat{\mathbf{x}}_{i+1}\right) \tag{29}$$
3) a_i, distances between origins along the common normal,
$$a_i = (\mathbf{p}_{i+1} - \mathbf{p}_i) \cdot \hat{\mathbf{x}}_{i+1} \tag{30}$$
4) α_i, angle between the previous to the current z-axis around the common normal,
$$\alpha_i = \operatorname{atan2}\left((\hat{\mathbf{z}}_i \times \hat{\mathbf{z}}_{i+1}) \cdot \hat{\mathbf{x}}_i, \hat{\mathbf{z}}_i \cdot \hat{\mathbf{z}}_{i+1}\right). \tag{31}$$

IV. Experiment and Results

The algorithm was first tested and validated in V-REP (Coppelia Robotics GmbH, Zürich, Switzerland), a robotics simulator. With an extensive library of robotic models available, it permitted testing the algorithm with different models containing varying kinematic structures, and joint types / dispositions in space. The proposed method was tested with the following virtual robot models: i) ABB IRB 140, 6-DoF multipurpose industrial robot; ii) Universal Robot UR3, 6-DoF collaborative robot; iii) KINOVA MICO, 6-DoF collaborative robot with a curved wrist; and the iv) Omron Cobra 650, 4-DoF SCARA robot.

The dimensions of each virtual model were verified against the available online documentation [14]–[17]. Tests were conducted in a dynamic simulation environment, meaning that each robot link has a physically modeled body and the joints connecting the links were driven by a dynamically simulated motor based on torque/force input and a PID-controller. These physical elements were handled by the Vortex physical engine (CM Labs, Montreal, QC, Canada). The physics engine inherently introduces noise to the read joint and end-effector positions, adding to the simulation realism.

The proposed method was then tested in two real robotic systems, an LBR iiwa R800 (KUKA, Augsburg, Germany) and a Sawyer (Rethink Robotics - HAHN Group, Bergisch Gladbach, Germany). Both are lightweight anthropomorphic serial manipulators designed for sensitive and cooperative tasks, and tailored for research development.

The **KUKA LBR iiwa R800** robot fits in the SRS (spherical-revolute-spherical) category of redundant manipulators [18]. In each of its 7 revolute joints the robot includes torque sensors and position encoders. For real-time control and acquisition of the robotic manipulator joint positions, the module Connectivity Sunrise.FRI (*Fast Robotic Interface*) was used. A client application running in a remote host was developed to communication with the controller unit (Sunrise Cabinet) in a local network. Each 5ms the client forwards the target joint positions to be reached in the next control cycle, and receives the current joint positions, Fig. 4.

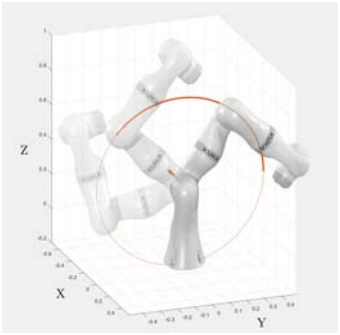

Fig. 4: Example of the robot motion that generates the tracked point trajectory for the second joint. The thicker lines represent the position of the tracked point acquired relative to the robot base frame. The thinner lines represent the best fitting circle to the trajectories.

The **Sawyer** robot is an anthropomorphic robot with 7 revolute joints that contrary to the LBR iiwa R800 does not have a SRS structure. This is due to due to several offsets distributed along its kinematic chain - 2 at the shoulder, 1 at the elbow and 1 at the wrist - which make it a particularly interesting case for DH parameter identification. The Intera SDK software interface was used to remotely communicate with the controller using the ROS API [19]. The client application receives the robot joint positions each 5ms.

The data acquisition process to determine the model parameters was similar across all robot models. The robots execute a series of position controlled movements defined in joint space, TABLE I. The sequence of motions proceeds iteratively from the first to the last joint as follows[1]:

1) the robot executes a PTP (point-to-point) movement to a set of initial joint positions, θ_{init};
2) the robot moves one joint to its starting position, $\theta_{i,\text{start}}$;
3) the position of $\mathbf{p}$ starts being acquired;
4) the robot drives the same joint until reaching its end position, $\theta_{i,\text{end}}$;

[1]See video at https://youtu.be/DUqfr_Q9n38

5) the point acquisition stops and the file is saved under the identification of the current joint.

The acquired points datasets files serve as the input for the proposed algorithm implemented in MATLAB[2], see an example in Fig. 5.

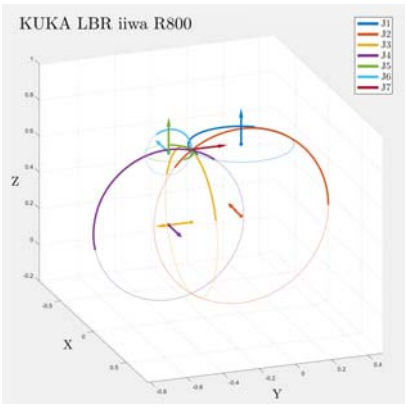

Fig. 5: Acquired point trajectories. For each revolute joint, the motion axis is represented at the center of rotation.

TABLE I: Initial, start and end joint coordinates used during the trajectory execution. For readibility, the joint positions are displayed in degrees, except for the Cobra's J_3 that is displayed in millimeters.

		J_1	J_2	J_3	J_4	J_5	J_6	J_7
ABB	θ_{init}	0	0	0	0	–	–	–
	$\theta_{i,\text{start}}$	-90	-60	-30	-90	-90	-90	–
	$\theta_{i,\text{end}}$	90	60	90	90	90	90	–
UR3	θ_{init}	0	0	0	0	–	–	–
	$\theta_{i,\text{start}}$	-90	-115	-145	-90	-90	-90	–
	$\theta_{i,\text{end}}$	90	0	0	90	90	90	–
MICO	θ_{init}	180	270	270	180	180	180	–
	$\theta_{i,\text{start}}$	90	180	90	90	90	90	–
	$\theta_{i,\text{end}}$	270	270	270	270	270	270	–
Cobra	θ_{init}	0	0	0*	0	–	–	–
	$\theta_{i,\text{start}}$	-115	-135	-200*	-160	–	–	–
	$\theta_{i,\text{end}}$	115	135	0*	160	–	–	–
KUKA	θ_{init}	90	-90	0	-90	0	90	0
	$\theta_{i,\text{start}}$	-35	-105	-70	-110	-70	-35	-70
	$\theta_{i,\text{end}}$	105	35	70	30	70	105	70
Sawyer	θ_{init}	90	-90	0	-90	0	90	0
	$\theta_{i,\text{start}}$	-35	-105	-70	-110	-70	-35	-70
	$\theta_{i,\text{end}}$	105	35	70	30	70	105	70

The proposed algorithm was tested for the robotic manipulators listed above. The measured DH parameters were compared to the nominal values and the results are listed in TABLE II. The parameters obtained have, for the most part, sub-millimetric and sub-degree deviation to the nominal values for the virtual and the real robotic manipulators. Two odd cases occur as a consequence of the undetermination of the common normals and intersection points in parallel joints of the MICO and the UR3. The value discrepancy is related to the d parameter, the offset from the previous z-axis to the common normal. These differences are related to the non-continuity of the DH parameters in actuators with parallel, or almost parallel consecutive joint axes. These discrepancies are nullified in the next algorithm iteration where the joint axis i and $i+1$ intersect, causing no impact in the Forward Kinematic calculation. Another issue occurred with the UR3 robot, the algorithm determined the common normal (x-axis) at the 5th joint to be the inverse of the nominal values. This issue is solved by providing the final frame (${}^{n-1}\mathbf{T}_n$) as suggested in subsection III-E.

Experiments were conducted with different ranges of joint trajectories acquired. It is noteworthy that trajectories where each joint traveled about a third of the total mechanical range returned parameters more similar to the nominal values. When the trajectory distance was increased beyond that point, no noticeable differences were reported. On the other hand, decreasing the traveled distance below a third of the total range lead to increasingly dissonant DH parameters from the nominal ones.

[2]Repository at https://github.com/neuebot/Kinematic-Calibration

V. Conclusion

A novel algorithm for automatic determination of the Standard Denavit-Hartenberg parameters of serial robotic manipulators was proposed. It determines a correct set of DH parameters for several robots with variable kinematic structures and joint types/dispositions. The method was tested in two real robots and four dynamically simulated robots, the later using the Vortex physical engine to reproduce as closely as possible the behavior of the real robots.

The applicability of the algorithm is twofold. First, if the end-effector position is determined using the robot controller, the intrinsic parameters used by the controller can be extracted as they are usually non-accessible. Second, if an external sensor (e.g. optical tracker) is used to measure the end-effector position, one can perform kinematic calibration within the error margin of the sensor.

The algorithm proved to be valid for the detection of DH parameters on all tested systems and yet it is affected by the limitations of the DH notation, i.e. the non-continuity in actuators with parallel, or almost parallel consecutive joints.

References

[1] J. Denavit and R. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Trans. ASME E, Journal of Applied Mechanics*, vol. 22, pp. 215–221, 1955.

[2] S. Hayati, "Robot arm geometric link parameter estimation," in *The 22nd IEEE Conference on Decision and Control*. IEEE, 1983, pp. 1477–1483.

[3] H. Stone, *Kinematic Modeling, Identification, and Control of Robotic Manipulators*, 1st ed., ser. The Kluwer International Series in Engineering and Computer Science. Boston, MA: Springer US, 1987, vol. 29.

[4] R. He, Y. Zhao, S. Yang, and S. Yang, "Kinematic-Parameter Identification for Serial-Robot Calibration Based on POE Formula," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 411–423, jun 2010.

[5] X. Yang, L. Wu, J. Li, and K. Chen, "A minimal kinematic model for serial robot calibration using POE formula," *Robotics and Computer-Integrated Manufacturing*, vol. 30, no. 3, pp. 326–334, jun 2014.

[6] W. Veitschegger and Chi-haur Wu, "A method for calibrating and compensating robot kinematic errors," in *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, vol. 4. Institute of Electrical and Electronics Engineers, 1987, pp. 39–44.

TABLE II: Denavit-Hartenberg parameters for the studied robots. The measured Denavit-Hartenberg parameters ($a_{\text{meas},i}$, $\alpha_{\text{meas},i}$, $d_{\text{meas},i}$ and $\theta_{\text{meas},i}$) and the differences to the nominal values ($a_{\text{error},i}$, $\alpha_{\text{error},i}$, $d_{\text{error},i}$ and $\theta_{\text{error},i}$) for each joint are presented. The a and d parameters are represented in millimeters, the α and θ in degrees. Values truncated at $1e-9$.

	i	$a_{\text{meas},i}$	$a_{\text{error},i}$	$\alpha_{\text{meas},i}$	$\alpha_{\text{error},i}$	$d_{\text{meas},i}$	$d_{\text{error},i}$	$\theta_{\text{meas},i}$	$\theta_{\text{error},i}$
ABB	1	70.0007	6.925e-04	90.0000	0.000e+00	352.1216	1.216e-01	0.0000	0.000e+00
	2	359.9906	-9.351e-03	0.0000	0.000e+00	-0.0525	-5.255e-02	90.0006	5.866e-04
	3	0.0622	6.223e-02	89.9765	-2.351e-02	-0.1640	-1.640e-01	0.0000	0.000e+00
	4	-0.0985	-9.850e-02	-89.9765	2.351e-02	379.9459	-5.412e-02	0.0000	0.000e+00
	5	0.0657	6.573e-02	89.9785	-2.153e-02	-0.0005	-4.775e-04	0.0000	0.000e+00
	6	-0.0112	-1.123e-02	0.0000	0.000e+00	65.0010	9.562e-04	0.0000	0.000e+00
UR3	1	0.0853	8.532e-02	-90.0221	-2.212e-02	152.0021	1.021e-01	0.0000	0.000e+00
	2	243.3269	-3.231e-01	0.0008	7.892e-04	112.3699	1.124e+02	0.1516	1.516e-01
	3	213.4347	1.847e-01	-0.0056	-5.630e-03	-0.0344	-3.438e-02	-0.0200	-1.996e-02
	4	-0.0218	-2.184e-02	-90.0030	-3.018e-03	0.0483	-1.123e+02	0.0000	0.000e+00
	5	0.0167	1.673e-02	-89.9977	-1.800e+02	85.3869	3.695e-02	-179.9813	-1.800e+02
	6	0.0000	0.000e+00	0.0000	0.000e+00	81.7164	-1.836e-01	0.0000	0.000e+00
MICO	1	-0.0306	-3.064e-02	-90.0016	-1.606e-03	275.4593	-4.069e-02	0.0000	0.000e+00
	2	290.0117	1.170e-02	179.9931	-6.909e-03	118.0541	1.181e+02	-89.9911	8.900e-03
	3	-0.0326	-3.256e-02	90.0137	1.369e-02	111.0306	1.180e+02	-90.0000	0.000e+00
	4	0.1422	1.422e-01	-59.9959	4.061e-03	-166.1171	-3.541e-02	0.0000	0.000e+00
	5	-0.1094	-1.094e-01	-60.0130	-1.303e-02	-85.4399	1.234e-01	0.0000	0.000e+00
	6	-0.0002	-2.317e-04	0.0000	0.000e+00	-42.7119	6.978e-02	0.0000	0.000e+00
Cobra	1	399.9868	-1.322e-02	0.0001	9.326e-05	204.9906	-9.373e-03	0.0012	1.200e-03
	2	249.9857	-1.428e-02	0.0274	2.737e-02	-0.0001	-6.623e-05	-0.0045	-4.510e-03
	3	0.0000	0.000e+00	-0.0355	-3.554e-02	0.0000	0.000e+00	0.0000	0.000e+00
	4	0.0253	2.532e-02	0.0000	0.000e+00	0.0091	9.134e-03	0.0000	0.000e+00
KUKA	1	0.0001	1.435e-04	-90.0002	-2.417e-04	339.9918	-8.188e-03	0.0001	1.259e-04
	2	0.0128	1.281e-02	90.0004	3.765e-04	0.0002	1.656e-04	-0.0011	-1.086e-03
	3	0.0002	2.023e-04	89.9993	-6.760e-04	399.9947	-5.270e-03	0.0000	0.000e+00
	4	0.0005	4.710e-04	-89.9997	3.212e-04	-0.0074	-7.380e-03	-0.0003	-3.020e-04
	5	0.0333	3.333e-02	-90.0002	-1.683e-04	399.9951	-4.854e-03	0.0000	0.000e+00
	6	-0.0107	-1.066e-02	89.9942	-5.790e-03	-0.0144	-1.444e-02	0.0008	7.840e-04
	7	0.0000	0.000e+00	0.0000	0.000e+00	126.0081	8.147e-03	0.0000	0.000e+00
Sawyer	1	80.2571	-7.429e-01	-89.8740	1.260e-01	316.2051	-7.949e-01	-0.1954	-1.954e-01
	2	0.0573	5.733e-02	89.9338	-6.624e-02	192.6113	1.113e-01	90.0000	0.000e+00
	3	-0.4088	-4.088e-01	-89.9007	9.927e-02	399.8220	-1.780e-01	0.0000	0.000e+00
	4	0.1744	1.744e-01	89.9926	-7.397e-03	-168.8476	-3.476e-01	0.0229	2.285e-02
	5	-0.7611	-7.611e-01	-90.0842	-8.416e-02	400.1897	1.897e-01	0.0000	0.000e+00
	6	-0.1449	-1.449e-01	89.6653	-3.347e-01	135.8429	-4.571e-01	-0.0396	-3.964e-02
	7	0.0000	0.000e+00	0.0000	0.000e+00	132.6210	-1.129e+00	0.0000	0.000e+00

[7] Y. Wu, A. Klimchik, S. Caro, B. Furet, and A. Pashkevich, "Geometric calibration of industrial robots using enhanced partial pose measurements and design of experiments," *Robotics and Computer-Integrated Manufacturing*, vol. 35, pp. 151–168, 2015.

[8] W. Khalil, S. Besnard, P. Lemoine, W. Khalil, S. Besnard, and P. Lemoine, "Comparison study of the geometric parameters calibration methods," *International Journal of Robotics and Automation*, vol. 15, no. 2, pp. 56–67, 2009.

[9] H. Hage, P. Bidaud, and N. Jardin, "Practical consideration on the identification of the kinematic parameters of the Stäubli TX90 robot," in *Worl d Congress in Mechanism and Machine Science*, 2011, pp. 19–25.

[10] J. Ketchel and P. Larochelle, "Collision detection of cylindrical rigid bodies for motion planning," in *Proceedings 2006 IEEE International Conference on Robotics and Automation.* IEEE, 2006, pp. 1530–1535.

[11] A. Hayat, R. Chittawadigi, A. Udai, and S. Saha, "Identification of Denavit-Hartenberg Parameters of an Industrial Robot," in *Proceedings of Conference on Advances In Robotics - AIR '13*. New York, New York, USA: ACM Press, 2013, pp. 1–6.

[12] C. Rajeevlochana, S. Subir, and K. Shivesh, "Automatic Extraction of DH Parameters of Serial Manipulators using Line Geometry," in *The 2nd Joint International Conference on Multibody System Dynamics*, Stuttgart, 2012, pp. 1–9.

[13] I. Fischer, *Dual-Number Methods in Kinematics, Statics and Dynamics*, 1st ed. CRC Press, 1998.

[14] ABB, "ABB IRB 140 Datasheet," pp. 1–2, 2019. [Online]. Available: https://new.abb.com/products/robotics/industrial-robots/irb-140/irb-140-data

[15] Universal Robots, "UR3 - Technical Details," p. 1, 2019.

[16] K. Robotics, "KINOVA MICO Robotic arm - User Guide," pp. 1–2, 2009. [Online]. Available: https://www.kinovarobotics.com/en/products/robotic-arms/kinova-gen2-ultra-lightweight-robot

[17] Omron, "Robotic Automation Industrial Robots Datasheets - Cobra 650," pp. 18–19, 2019. [Online]. Available: http://www.ia.omron.com/products/family/3733/download/catalog.html

[18] C. Faria, F. Ferreira, W. Erlhagen, S. Monteiro, and E. Bicho, "Position-based kinematics for 7-DoF serial manipulators with global configuration control, joint limit and singularity avoidance," *Mechanism and Machine Theory*, vol. 121, pp. 317–334, 2018.

[19] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," *ICRA workshop on open source software*, vol. 3, no. 3.2, p. 5, 2009.