

Article

Initial Estimation of Kinematic Structure of a Robotic Manipulator as an Input for Its Synthesis

Daniel Huczala ^{1,*}, Tomáš Kot ¹, Martin Pfurner ², Dominik Heczko ¹ and Petr Oščádal ¹
and Vladimír Mostýn ¹

¹ Department of Robotics, Faculty of Mechanical Engineering, VSB-TU of Ostrava, 17. Listopadu 2172/15, 708 00 Ostrava-Poruba, Czech Republic; tomas.kot@vsb.cz (T.K.); dominik.heczko@vsb.cz (D.H.); petr.oscandal@vsb.cz (P.O.); vladimir.mostyn@vsb.cz (V.M.)

² Unit of Geometry and Surveying, University of Innsbruck, Technikerstr. 13, A-6020 Innsbruck, Austria; martin.pfurner@uibk.ac.at

* Correspondence: daniel.huczala@vsb.cz

Abstract: Researchers often deal with the synthesis of the kinematic structure of a robotic manipulator to determine the optimal manipulator for a given task. This approach can lower the cost of the manipulator and allow it to achieve poses that might be unreachable by universal manipulators in an existing constrained environment. Numerical methods are broadly used to find the optimum design but they often require an estimated initial kinematic structure as input, especially if local-optimum-search algorithms are used. This paper presents four different algorithms for such an estimation using the standard Denavit–Hartenberg convention. Two of the algorithms are able to reach a given position and the other two can reach both position and orientation using Bézier splines approximation and vector algebra. The results are demonstrated with three chosen example poses and are evaluated by measuring manipulability and the total link length of the final kinematic structures.

Keywords: manipulator design; robot kinematics; synthesis of kinematic structure



Citation: Huczala, D.; Kot, T.; Pfurner, M.; Heczko, D.; Oščádal, P.; Mostýn, V. Initial Estimation of Kinematic Structure of a Robotic Manipulator as an Input for Its Synthesis. *Appl. Sci.* **2021**, *11*, 3548. <https://doi.org/10.3390/app11083548>

Academic Editor: Giuseppe Carbone

Received: 19 March 2021

Accepted: 13 April 2021

Published: 15 April 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Nowadays, manufacturing industry, the most common type of robotic manipulator has six serial axes (degrees of freedom—DoFs) that are arranged in a so-called universal kinematic structure, e.g., the robots of ABB, KUKA, Fanuc, Yaskawa, and many others. In a very simplified way, the typical process for the deployment of a robot is to analyse the manufacturing process and workplace area first, followed by the choosing of a universal robotic manipulator and simulating the process. If the manipulator can reach the desired poses and fulfil the given task, further deployment can be considered.

However, using this universal kinematic structure is not always necessary, when a manipulator with fewer axes is suitable to perform the given task, or even possible, if the universal manipulator can face unavoidable collisions in an already existing environment. Additionally, they might not fulfil the desired advanced operation conditions, such as manipulability [1] and kinematic reliability [2]. Therefore, researchers are focused on the topic of the synthesis of the kinematic structure of manipulators, which means finding such a kinematic structure that is optimal for a given task. This approach of deployment of highly customised manipulators may lead to benefits like lowered energy consumption, accelerated manufacturing process cycles, or deploying manipulators in highly dense-built workplaces. An example of such a general structure is presented by Brandstötter et al., who delivered the so-called curved manipulator (CuMa) [3] with possible modifications of its structure during the operational process [4]. This is achieved by changing the temperature in the links so they become flexible. A different approach to a deformable manipulator was taken by Xu et al. [5], where the links are composed of a few components and it is possible to change the orientation between them. Clark et al. [6] uses air pressure to change the

kinematic structure of the presented malleable robot. The custom design of manipulators is desired not only in the manufacturing industry, but for example, in healthcare for helping with human upper limb rehabilitation [7] and shoulder joint rehabilitation [8].

There are two approaches to the synthesis of the kinematic structure of a manipulator. Analytically, it was solved by Hauenstein et al. [9] in the synthesis of three-revolute spatial chain for five poses. However, once a path becomes more complex and requires more degrees of freedom (more manipulator axes), the numerical approach is applied utilising optimisation algorithms. Among them, evolutionary robotics with genetic algorithms (GA) are probably the most common. Chocron et al. [10] used an adaptive multi-chromosome evolutionary algorithm to build a modular manipulator. Furthermore, GA can be used to build a manipulator from adaptive modules to perform a desired task [11]. Pastor et al. [12] compared straight, rounded, and curved mechanism links synthesised using GA. Valsamos et al. created so-called pseudo-joints (links which can be modified) and proposed a GA that tries to find the kinematic structure with the best manipulability [13]. It was later verified in an experiment with a real manipulator [14]. The synthesis of a parallel manipulator is addressed in [15]. As an example of non-industrial application, the work by Zeiaee et al. [16] deals with the optimisation of an eight-DoFs upper-limb exoskeleton.

In addition to GAs, the global optimum of an objective function can be searched with Simulated Annealing algorithm [17] or by a heuristic-guided tree search algorithm [18]. Another numerical method for finding the optimal manipulator for a given task is to search for a local minimum of an objective function. To solve this, a nonlinear programming (NLP) can be used, as it was implemented by Dogra et al. [19] for the design of a modular manipulator. In the paper, an optimal kinematic structure was proposed based on the minimisation of the joint torques. Another usage of NLP is trying to find an optimal design minimizing the path length in joint space [20].

The results of the already described papers [19,20] seem promising in their application of nonlinear programming to solve task-based custom manipulator design, however, there is one unanswered question in their work. In general, nonlinear programming traditionally requires that a starting point is given as part of the problem data, and comparative numerical testing is done using these traditional starting points [21]. In the case of robotic manipulators, if there is a given random path for a robotic manipulator, how should the starting point (initial values, initial estimation) of its kinematic structure look like?

The work [19] mentions a set of input values without any detailed explanation of how those values were determined. In [20], eight initial seeds are applied, but they are random values, which might be a cause of why no solution was found at all in some cases. In [17], they searched for a global optimum; however, the input values are also random, which may extend the time of solving the objective function. Even in a book by Ghafil et al. [22] which serves as an introduction to the optimisation of kinematic structures, the initial values for all described methods are obtained randomly without any detailed discussion. Therefore, a possible answer to the previously stated question will be addressed in this paper using multiple approaches.

In this paper, a geometric analysis to estimate kinematic structures and related calculations are proposed and discussed. The outcomes may avoid relying on randomness, which in the case of local-optimum-search algorithms may more frequently lead to convergence, making them reliable but still much faster than global-optimum-search methods. Moreover, the results can also be used in the previously mentioned algorithms using GAs, where they can serve as an optional first generation input, and in global-optimum-search algorithms, where they can reduce the time needed for the optimisation. In addition to that, the procedure may serve as an input for other custom manipulator design challenges such as collision avoidance [23].

A Denavit–Hartenberg notation [24] (standard DH parameters) was applied in the presented algorithms to obtain a general kinematic structure of a robot. The DH parameters were widely used, however, they also bring some disadvantages in the case of general structures. These are discussed later. Some algorithms behind the automatic placement of

DH parameters have already been presented. In [25], a vector-algebra is applied to extract the parameters. Corke [26] creates a string of elementary translations and rotations from the user-defined base coordinate to the end-effector and factorises the string afterwards. An approach by Rajeevlochana et al. [27] is using line geometry to obtain the parameters. In addition to that, some researchers proposed their algorithms and they also identified (and verified) the DH parameters with an external sensor device in simulation [28] or experiments [29]. However, all of these algorithms were applied to already existing robots or similar devices. On the other hand, the algorithms presented in this paper are here to determine and “create” a non-existing manipulator (its kinematic structure) that can achieve a freely given pose.

2. Materials and Methods

When an algorithm is searching for a local minima of a function, the results may differ upon the choice of initial values. To find the optimal solution for two initial values with different outcomes, the cost function has to be compared afterwards. More accurate initial values can lead to fewer iterations that the algorithm needs, which can also save computing time. In this section, four algorithms of automatic assignment of DH parameters are presented, so they can serve as initial values for the synthesis of kinematic structure. The inputs are the position of the base of a robot, its tool-center point (TCP) pose, and the number of joints.

In this paper, we denote the transformation matrix between two frames as \mathbf{J} with axis vectors and position coordinates as shown in Equation (1). The \vec{n} (normal) is the X axis vector, the \vec{o} (orientation) is Y axis vector, the \vec{a} (approach) is Z axis vector, and the \vec{t} (translation) is the position coordinate vector of a pose. \mathbf{J} is a special Euclidean group of rigid body displacements in three dimensions (SE3) representing 3D rigid-body motion:

$$\mathbf{J} = \begin{bmatrix} \vec{n} & \vec{o} & \vec{a} & \vec{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

We also use unit vectors of the X, Y, and Z axes. They are denoted as \vec{i} , \vec{j} , and \vec{k} :

$$\vec{i} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}; \quad \vec{j} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}; \quad \vec{k} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (2)$$

For visualisation and work with SE3 groups, we used the MATLAB[®], and Robotics Toolbox that was made by P. Corke. It is described in his book [30] and accessible as open source Github repository [31].

DH parameters are the most suitable and easily applicable technique for kinematic structures that have parallel or orthogonal axes. The typical procedure is that one has a robot and places the coordinate frames following the convention [24]. However, what to do when there is a given pose that is needed to be reached while no robot is chosen yet? This is the problem for the synthesis of kinematic structure. There is one important issue related to DH parameters. Between two general poses (right-hand rule following coordinate frames), it is uncertain if the transformation matrix $\mathbf{J}_{i-1,i}$ from $(i-1)$ th pose to i th pose can be achieved following the typical procedure as the multiplication of a rotation matrix of θ_i around the z_{i-1} axis, the translation matrix of d_i along the z_{i-1} axis, the translation matrix of a_i along the x_i axis, and the rotation matrix of α_i around the x_i axis, as shown in the following equations:

$$\text{Rot}(z_{i-1}, \theta_i) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$\text{Trans}(z_{i-1}, d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$\text{Trans}(x_i, a_i) = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$\text{Rot}(x_i, \alpha_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

$$\mathbf{J}_{i-1,i} = \text{Rot}(z_{i-1}, \theta_i) \text{Trans}(z_{i-1}, d_i) \text{Trans}(x_i, a_i) \text{Rot}(x_i, \alpha) \quad (7)$$

It is clear that rotation and translation around and along the y axis are missing to achieve all possible poses. In DH convention, it is mitigated by placing the joint coordinate frames in a specific way and applying Equation (7), as explained in [24]. However, if there are two general poses $(i-1)$ th and i th which do not follow DH convention, it is possible to find a common perpendicular between their two z_{i-1} and z_i axes. Please see Figure 1. Rotation around z_{i-1} to the direction of the perpendicular is θ_i . d_i is the distance from the x_{i-1} axis to the intersection point P of the perpendicular and z_{i-1} axis. The distance along the perpendicular is equal to the a_i distance between these two frames. Finally, the rotation around the x_i axis to the direction of z_i is α_i . If another displacement of d_{i+1} is added and a rotation θ_{i+1} is applied, the previously unreachable (by four DH parameters) general pose i th becomes achievable by four DH parameters of $(i-1)$ th joint and two DH parameters d_{i+1} and θ_{i+1} of the i th joint. It can also represent an end-effector coordinate frame if the $(i-1)$ th joint was the last joint. This approach is presented in detail in [27,29]. For the following calculations, we enhanced a script made by Brodsky [32] to find a common perpendicular and intersection points P and Q . This can be found in the Supplementary Material of this paper.

We used 3 poses to demonstrate the strong and weak points of the four presented algorithms to synthesise manipulators guiding their end-effector through the given position or pose, so everyone can choose the right solution for its implementation. They are also compared in Section 3 by manipulability and arm length. The first pose is a general one. The second pose is also general, but with a small offset between its Z axis and the Z axis of the base frame. The third pose has the parallel Z axis with the base Z axis. The poses are visualised in Figure 2:

$$\text{Pose}(1) = \begin{bmatrix} -0.50 & -0.18 & -0.84 & -1.0 \\ -0.06 & 0.98 & -0.17 & 0.9 \\ 0.86 & -0.02 & -0.50 & 0.8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

$$\text{Pose}(2) = \begin{bmatrix} 0.81 & -0.34 & -0.46 & -0.4 \\ 0.48 & -0.02 & 0.87 & 0.6 \\ -0.30 & -0.93 & 0.14 & 0.7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

$$\text{Pose}(3) = \begin{bmatrix} 0 & 1 & 0 & 0.4 \\ 1 & 0 & 0 & 0.6 \\ 0 & 0 & -1 & 0.7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

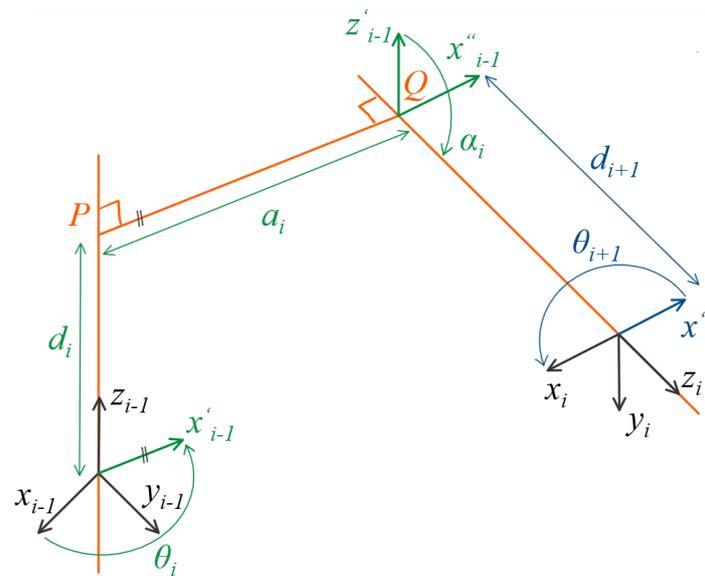


Figure 1. The principle of obtaining DH parameters between two general poses. Standard DH parameters are green; additional parameters are marked with blue colour.

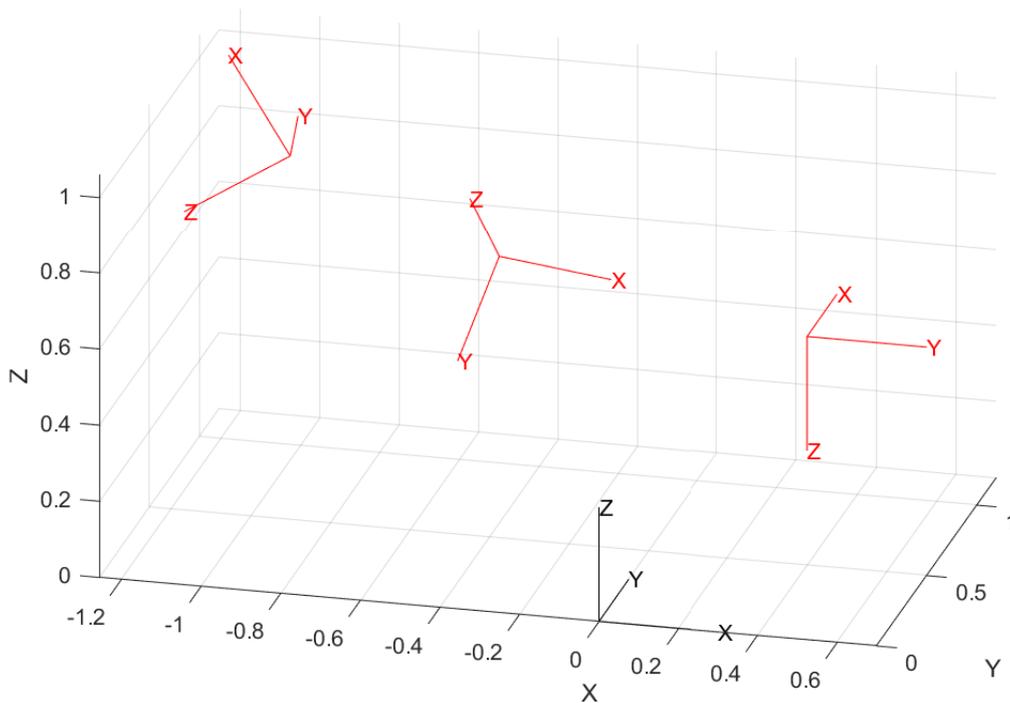


Figure 2. The poses (red) chosen for the demonstration of the working principle. Base coordinate frame has a black colour. Pose(1) is on the left, Pose(2) is in the middle, and Pose(3) is on the right.

Two of the four presented algorithms utilised Bézier curves (splines) which are easy to implement between two given coordinate frames. For the presented calculations, only four control points are required to define a Bézier curve. We used a script by Bai [33] to calculate the curve. The control points P_{1-4} were calculated using the following equations, where \vec{t}_b is the base point coordinate, \vec{t}_p is the pose point coordinate, \vec{a}_b is the rotational vector of the Z axis of the base, \vec{a}_p is the rotational vector of the Z axis of the pose, and p is

the parameter related to the Euclidean distance between the base and the pose. The Bézier splines are visualised in Figure 3 for the chosen poses.

$$P_1 = \vec{t}_b \quad (11)$$

$$P_2 = \vec{t}_b + p\vec{a}_b \quad (12)$$

$$P_3 = \vec{t}_p - p\vec{a}_p \quad (13)$$

$$P_4 = \vec{t}_p \quad (14)$$

$$p = \frac{\|\vec{t}_p - \vec{t}_b\|}{2} \quad (15)$$

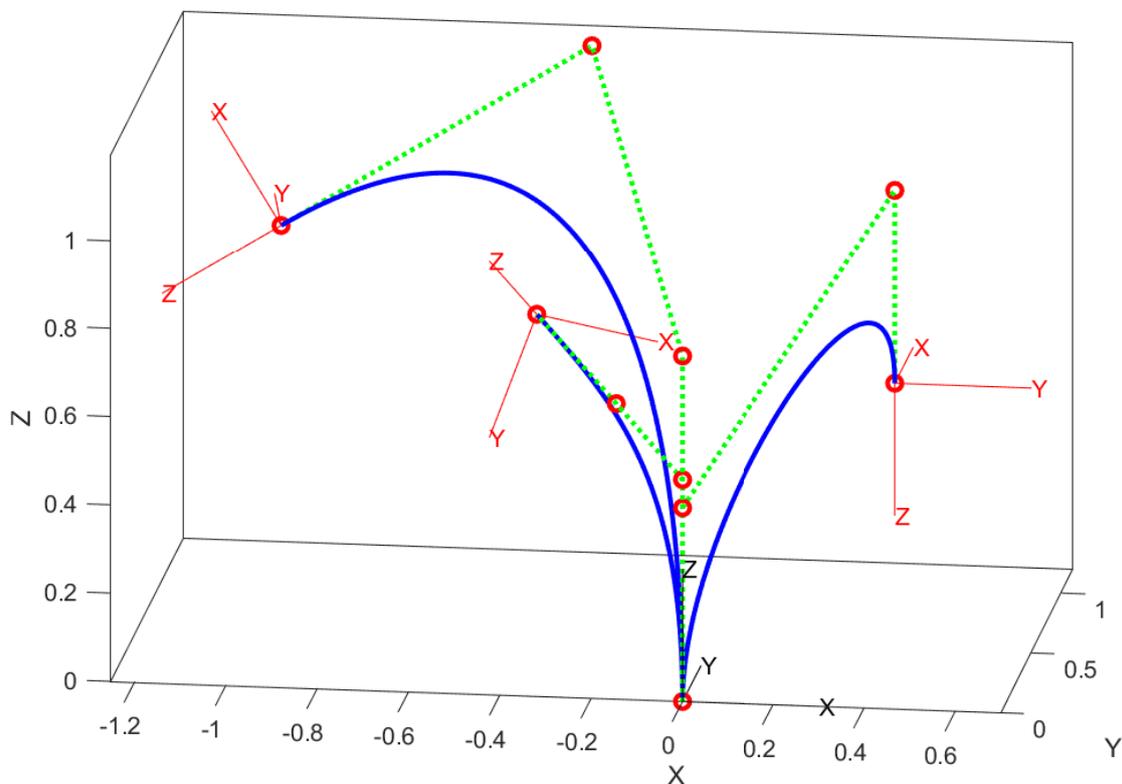


Figure 3. Bézier splines (blue curves) between the poses and base; control points are shown as red circles.

The generated kinematic structures that have served as examples in this paper have 4 joints; however, the presented algorithms are general and can provide a solution from 3 to an unlimited number of joints. In the following subsections, all 4 procedures are presented. The types A and B only deal with the position (translational part) of a given pose, so they do not fulfil the given orientation. However, this might be enough in some cases. The other two types C and D are able to achieve a pose including orientation using the common perpendicular approach, but in some specific poses it generates structures with joints in collision. The A and C types are obtained using vector algebra only, and the B and D types use Bézier's curve approximation.

Three variables are input for all presented algorithms. It is the transformation of the robot base, the transformation of the TCP pose, both with respect to the world coordinate frame, and the desired number of joints. In our case, the base is an identity matrix. We used 3 transformations of the poses presented before, and the number of joints is four, as already said.

2.1. Type A—The Nearest Distance to Achieve a Position

This simple structure is obtained by finding the distance between the poses projected into the XY base plane. Only a positional vector of a pose is reached while the orientation is not taken into account. The implementation of such a structure is easy and straightforward. The idea is presented in Figure 4:

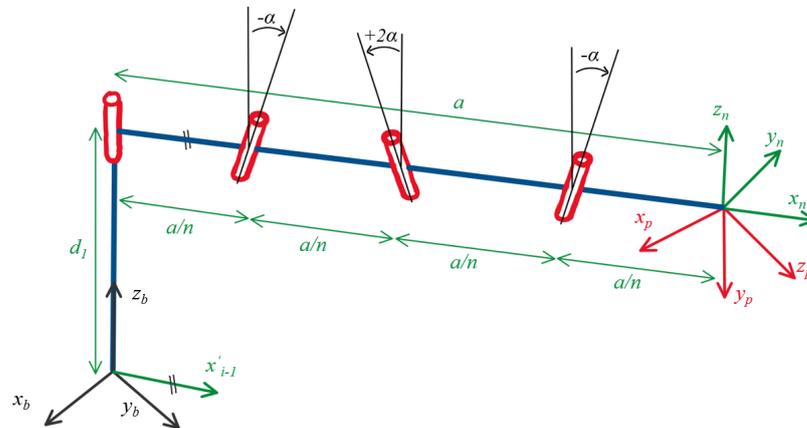


Figure 4. The schematic of type A estimation; the base frame is black, the pose frame is red, and the end-effector frame is green and does not fulfil the orientation of the pose.

The first step is to find the length of $a_{0..n}$ —the distance between joints in the X axis direction. The length is the projection of \vec{t}_p , the pose position vector, in the XY plane of the base, so only the X and Y coordinates are applied in Equation (16). n is the number of joints:

$$a_{1..n} = \frac{\sqrt{(\vec{t}_{p,x} - \vec{t}_{b,x})^2 + (\vec{t}_{p,y} - \vec{t}_{b,y})^2}}{n} \tag{16}$$

Then, find the length of d_1 —the offset of the joint along the Z axis. Only the Z axis coordinates of the two position vectors are applied:

$$d_1 = \vec{t}_{p,z} - \vec{t}_{b,z} \tag{17}$$

The $\theta_{1..n}$ are joint variables, and their offset is set to 0 degrees. The other parameters, $d_{2..n}$ and $\alpha_{1..n}$ can be set either to zero or they can be freely defined as \pm values, for example. We chose $\alpha_1 = \pi/4$, $\alpha_2 = -\pi/4$, etc. One must be careful in the case of an even/odd number of joints—the sum of such tweaks needs to be equal to zero.

2.2. Type B—Joints on Bézier Curve to Achieve a Position

This method places joints between the base and the pose on a Bézier curve. To be able to obtain DH parameters, the proposed algorithm is orienting the $(i - 1)$ th joint (its rotational matrix) in a way that the i th joint lies in the XZ plane of the previous joint. The schematic is shown in Figure 5:

Using Equations (11)–(15), the Bézier spline is approximated between a given base and a pose. The number of approximated points is equal to the number of joints n . $Q_{1..n}$ is the set of these points—coordinates of each point with respect to the base frame.

Let us define a set $J_{1..n}$ of SE3 objects representing the translation and orientation of the joints in the manipulator’s default position. As a first step, all $J_{1..n}$ are set to be equal to the given base (in our case, an identity matrix). We also define an SE3 object J_{n+1} representing the given end-effector pose. Now, for joints $J_{2..n}$, the following procedure is done.

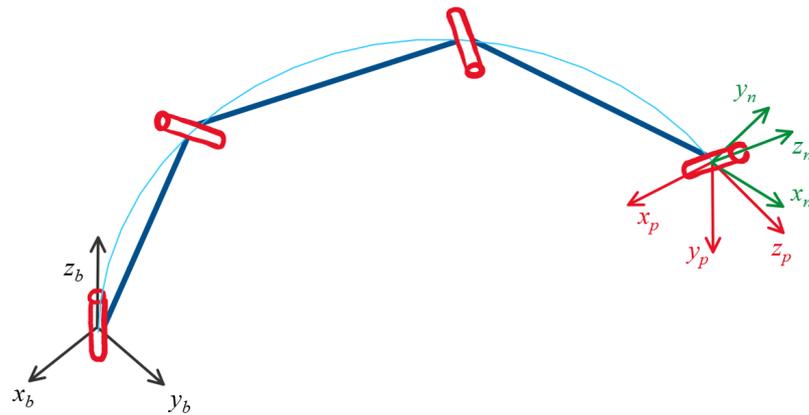


Figure 5. The schematic of type B estimation; the base frame is black, the pose frame is red, and the end-effector frame is green and does not fulfil the orientation of the pose. Bézier spline is shown as a light blue curve.

The i th joint is equal to the previous $(i - 1)$ th joint:

$$J_i = J_{i-1} \tag{18}$$

The frame J_i is translated on the Bézier curve changing its translation vector \vec{t}_i :

$$\vec{t}_i = Q_i \tag{19}$$

The position vector \vec{t}_i of J_i is expressed in the coordinate frame of the previous J_{i-1} using its inverse matrix:

$$\vec{t}'_i = J_{i-1}^{-1} \vec{t}_i \tag{20}$$

A projection of the \vec{t}'_i vector in the XY plane of the J_{i-1} frame is determined:

$$\vec{t}''_i = \vec{t}'_i - \begin{bmatrix} 0 \\ 0 \\ \vec{t}'_{i,z} \end{bmatrix} \tag{21}$$

The angle θ_i (DH parameter) between the unit vector \vec{i} of the J_{i-1} frame and the projection of the \vec{t}'_i vector is calculated as

$$\theta_i = \tan^{-1} \left(\frac{\|\vec{i} \times \vec{t}''_i\|}{\vec{i} \cdot \vec{t}''_i} \right) \tag{22}$$

While using a right-handed coordinate frame, it is necessary to check if an angle is rotating around an axis in the positive (counter clockwise) or negative (clockwise) direction. To determine this, we used the projection property of the dot product between the two vectors. In this case, if the dot product of the X_i axis is in the negative direction of the Y_{i-1} axis, the angle θ_i has to be multiplied by -1 :

$$\theta_i = \begin{cases} -\theta_i, & \text{if } \vec{j} \cdot \vec{t}''_i < 0 \\ \theta_i, & \text{otherwise} \end{cases} \tag{23}$$

Now, J_i can be updated using the matrix multiplication:

$$J_i = J_i \text{Rot}(z_{i-1}, \theta_i) \tag{24}$$

Thanks to the known translations, i and a_i , between the frames (from the Bézier curve approximation), and the calculated θ_i , only the angle α_i is missing among the DH parameters. The steps to determine it are similar as in the case of θ . Following the DH convention, the i th and $(i + 1)$ th frames are involved.

The position vector of J_{i+1} on the Bézier curve is given:

$$\vec{t}_{i+1} = Q_{i+1} \tag{25}$$

Position vector \vec{t}_{i+1} of J_{i+1} is expressed in the coordinate frame of the currently determining frame J_i using its inverse matrix:

$$\vec{t}'_{i+1} = J_i^{-1} \vec{t}_{i+1} \tag{26}$$

A projection of the \vec{t}'_{i+1} vector in the YZ plane of the J_i frame is calculated:

$$\vec{t}''_{i+1} = \vec{t}'_{i+1} - \begin{bmatrix} \vec{t}'_{i+1,x} \\ 0 \\ 0 \end{bmatrix} \tag{27}$$

The angle α_i is between the unit vector \vec{k} of the J_i frame and the projection of the \vec{t}'_{i+1} vector, calculated as the inverse tangent fraction of the cross and dot products of those two vectors:

$$\alpha_i = \tan^{-1} \left(\frac{\|\vec{k} \times \vec{t}''_{i+1}\|}{\vec{k} \cdot \vec{t}''_{i+1}} \right) \tag{28}$$

Using a right-hand rule for coordinate frames, if the dot product of the Z_i axis is in the negative direction of the Y_{i+1} axis, the angle θ_i has to be multiplied by -1 . Y_{i+1} is calculated as a cross product of \vec{t}''_{i+1} and \vec{i} vectors:

$$\alpha_i = \begin{cases} -\alpha_i, & \text{if } (\vec{t}''_{i+1} \times \vec{i}) \cdot \vec{k} < 0 \\ \alpha_i, & \text{otherwise} \end{cases} \tag{29}$$

Now, the final form of J_i that fulfils the DH convention between $(i - 1)$ th and i th can be obtained:

$$J_i = J_i \text{Rot}(x_i, \alpha_i) \tag{30}$$

This procedure works smoothly for all joints. However, it will probably not be possible to obtain such DH parameters between the last joint J_n and the given pose J_{n+1} to reach the pose with the right orientation. Therefore, this algorithm is extended as the type D estimation in Section 2.4.

2.3. Type C—Achieving a Pose with Common Perpendicular

This algorithm finds a common perpendicular between the Z axis of the base and the Z axis of the pose. The joints are placed on this perpendicular line, and the last joint is oriented in the direction of the Z axis. Both the position and orientation can be achieved using this approach, as Figure 6 shows.

At first, a common perpendicular and intersection points P and Q are determined between the Z_b and Z_p axes using the script made by Brodsky [32]. However, his algorithm was not providing good results if the 2 lines were parallel, so we enhanced it and added some functionality to mitigate this issue.

The next step is to calculate the angle α_{sum} between those two axes:

$$\alpha_{sum} = \tan^{-1} \left(\frac{\|\vec{a}_b \times \vec{a}_p\|}{\vec{a}_b \cdot \vec{a}_p} \right) \tag{31}$$

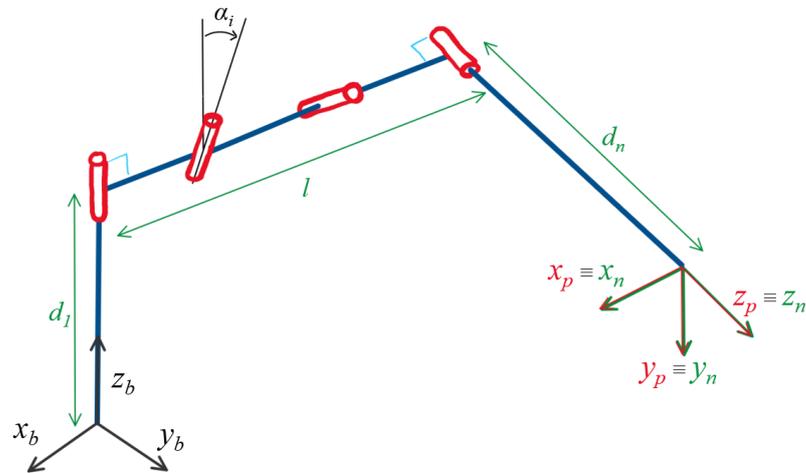


Figure 6. The schematic of the type C estimation; the base frame is black, the pose frame is red, and the end-effector frame is green and coincident with the pose.

Again, when using a right-handed coordinate system, it is necessary to determine whether the angle is positive or negative. We check the pose Z_{i+1} axis as a projection in the base Y_i axis:

$$\alpha_{sum} = \begin{cases} -\alpha_{sum}, & \text{if } \vec{0}_b \cdot \vec{a}_p < 0 \\ \alpha_{sum}, & \text{otherwise} \end{cases} \quad (32)$$

$\alpha_{1..n}$ is the angle between the Z axes of particular joints:

$$\alpha_{1..n-1} = \frac{\alpha_{sum}}{n-1} \quad (33)$$

$$\alpha_n = 0 \quad (34)$$

Now, we can calculate the rest of the DH parameters. $a_{1..n}$ is the distance between the joints. l is the length of a common perpendicular, and n is the number of joints:

$$a_{1..n-1} = \frac{l}{n-1} \quad (35)$$

$$a_n = 0 \quad (36)$$

d_0 is the distance from the base coordinate frame to the P-intersection point of the \vec{a}_b and common perpendicular. d_n is the distance from the Q, the intersection point of the \vec{a}_p and a common perpendicular, to the pose coordinate frame. d_n is also the translation of the end-effector from the last joint:

$$d_0 = (P - \vec{t}_b) \cdot \vec{a}_b \quad (37)$$

$$d_{1..n-1} = 0 \quad (38)$$

$$d_n = (\vec{t}_p - Q) \cdot \vec{a}_p \quad (39)$$

2.4. Type D—Joints on Bézier Curve while the Last Lies on Common Perpendicular to Achieve a Pose

This method extends type B estimation by adding a common perpendicular approach, used in type C, between the two last joints. This assures reaching the pose including orientation, as shown in Figure 7. The beginning steps are the same as in type C, but only from J_1 to J_{n-1} . The transformation of the last joint is determined using the following procedure.

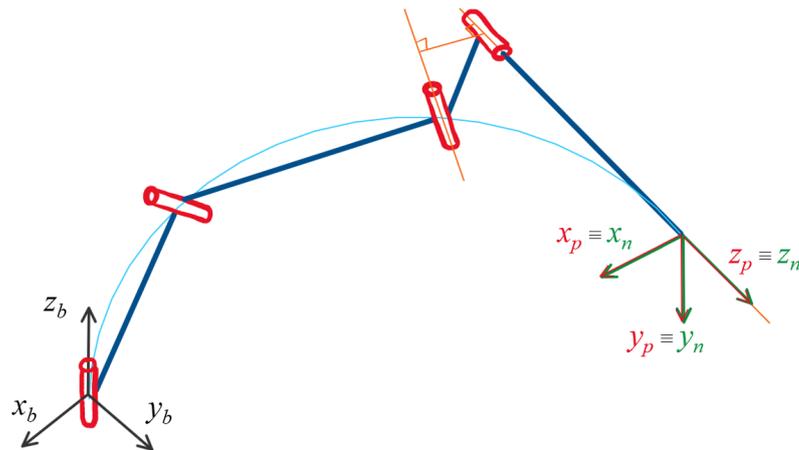


Figure 7. The schematic of the type D estimation; the base frame is black, the pose frame is red, the end-effector frame is green and coincident with the pose. Bézier spline is shown as a light blue curve.

Again, a common perpendicular and the intersection points P and Q are found between the joint J_{n-1} and the TCP pose J_{n+1} using the already presented ways. J_n is set equal to J_{n-1} :

$$J_n = J_{n-1} \tag{40}$$

Angle θ_n between X_{n-1} and X_n axes is obtained:

$$\theta_n = \tan^{-1} \left(\frac{\|\vec{n}_{n-1} \times \vec{PQ}\|}{\vec{n}_{n-1} \cdot \vec{PQ}} \right) \tag{41}$$

Right-hand rule check is performed:

$$\theta_n = \begin{cases} -\theta_n, & \text{if } \vec{\sigma}_{n-1} \cdot \vec{PQ} < 0 \\ \theta_n, & \text{otherwise} \end{cases} \tag{42}$$

J_n is rotated around Z_{n-1} axis afterwards:

$$J_n = J_n \text{Rot}(z_{n-1}, \theta_n) \tag{43}$$

Translation of the J_n along the Z_n and X_n is obtained by changing its translational vector \vec{t}_n , and it is equal to the coordinates of point Q :

$$\vec{t}_n = Q \tag{44}$$

Angle α_n between Z_n and Z_{n+1} axes is calculated:

$$\alpha_n = \tan^{-1} \left(\frac{\|\vec{a}_n \times \vec{a}_{n+1}\|}{\vec{a}_n \cdot \vec{a}_{n+1}} \right) \tag{45}$$

Right-hand rule check, if the dot product of Z_{i+1} axis is in the positive direction of the Y_i axis, the angle α_n has to be multiplied by -1 :

$$\alpha_n = \begin{cases} -\alpha_n, & \text{if } \vec{\sigma}_n \cdot \vec{a}_{n+1} > 0 \\ \alpha_n, & \text{otherwise} \end{cases} \tag{46}$$

The final transformation matrix of the last joint J_n is obtained:

$$J_n = J_n \text{Rot}(x_n, \alpha_n) \tag{47}$$

From this point, when all frames $J_{1..n}$ representing the joints are known, it is possible to derive the DH parameters between them.

3. Results

This section presents the kinematic structures generated by the presented algorithms for the three poses defined earlier (Figure 2). The results are discussed and later compared by manipulability measure and manipulator length. A table with calculated DH parameters is also included. The visualisation of the final kinematic structures is shown in Figures 8–10.

The types of estimation A and B are not able to reach a pose in terms of orientation in general; however, in some cases (as shown in Figure 10b) the real solution was found for the type B. In addition, if compared with a similar D result (Figure 10d) for *Pose(3)*, solution B provides shorter links. Furthermore, the A and B types are generated in a way where no collision of joints should occur.

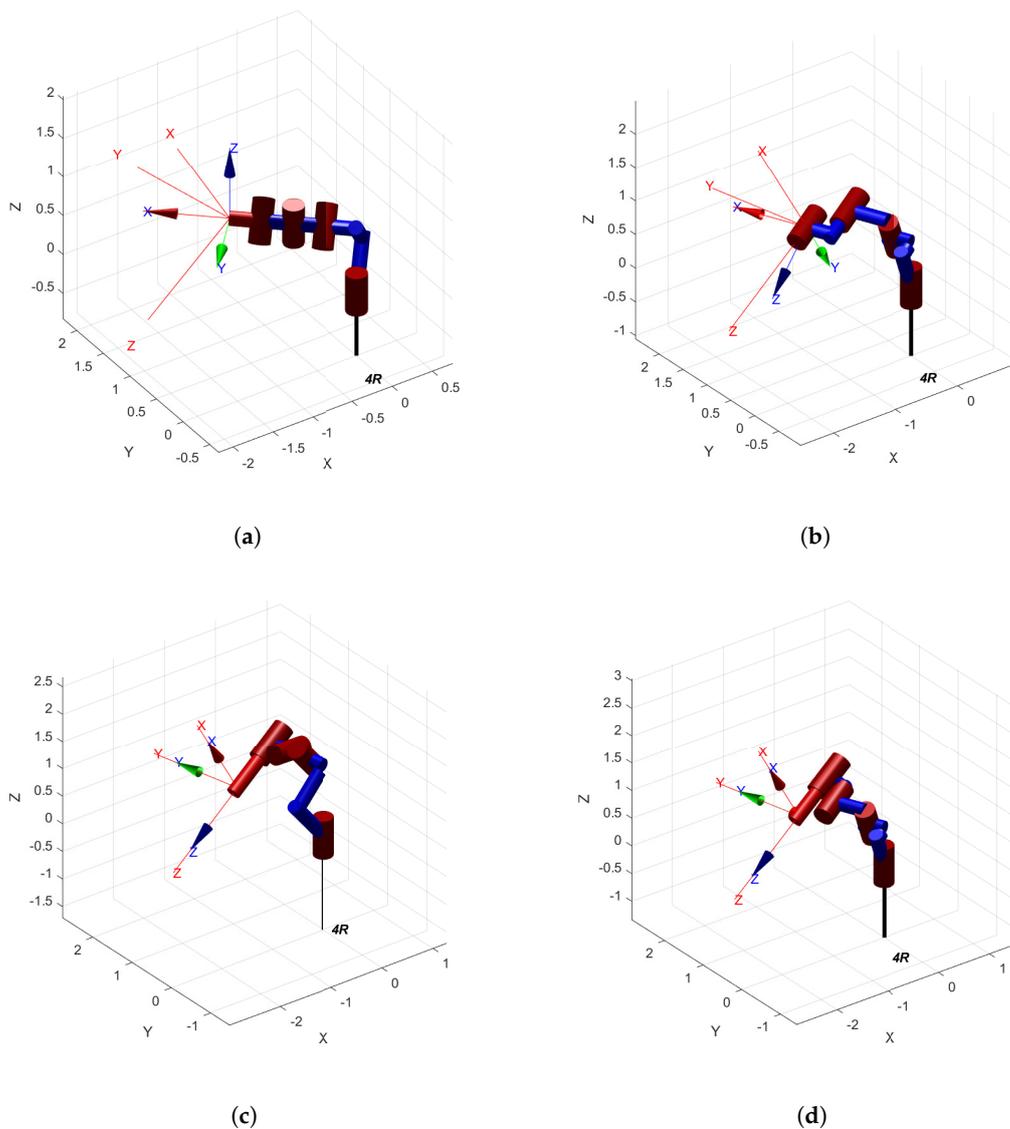


Figure 8. Initial estimation results for Pose (1): (a) type A estimation; (b) type B estimation; (c) type C estimation; and (d) type D estimation.

The type C may perform very well if the Z axes of the base and pose are parallel, as shown in Figure 10c; on the other hand, if the axes are very close to each other (the perpendicular distance is short), the joints are in collision, as shown in Figure 9c.

Placing joints on an approximated spline provides the most general result of the provided algorithms, as shown in Figures 8d and 9d, but it is struggling with parallel axes—see Figure 10d. This could be mitigated by tuning the Bézier curve driven point related to the pose and placing the joints not in a plane that is defined by the two parallel Z axes.

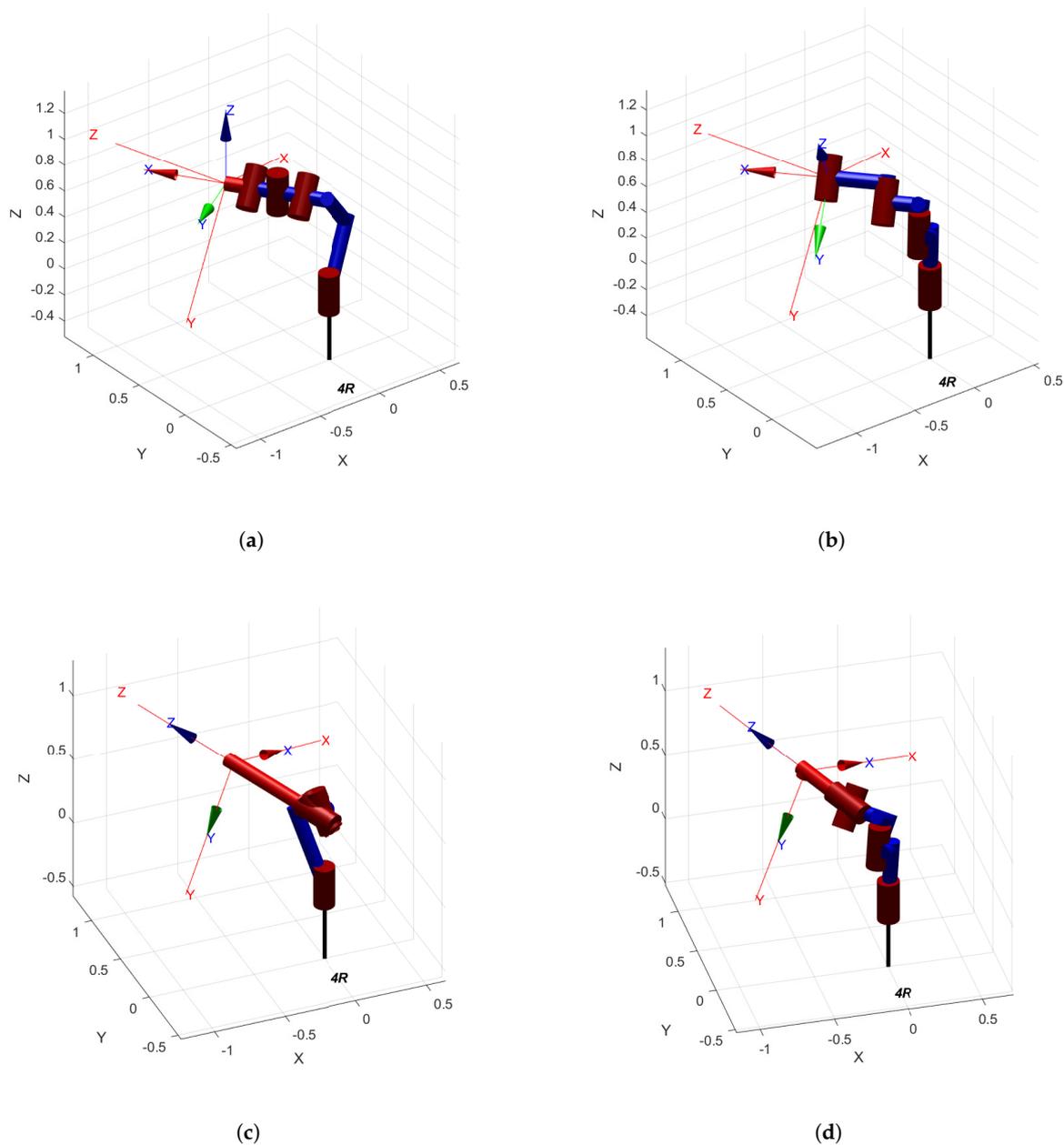


Figure 9. Initial estimation results for Pose (2): (a) type A estimation; (b) type B estimation; (c) type C estimation; and (d) type D estimation.

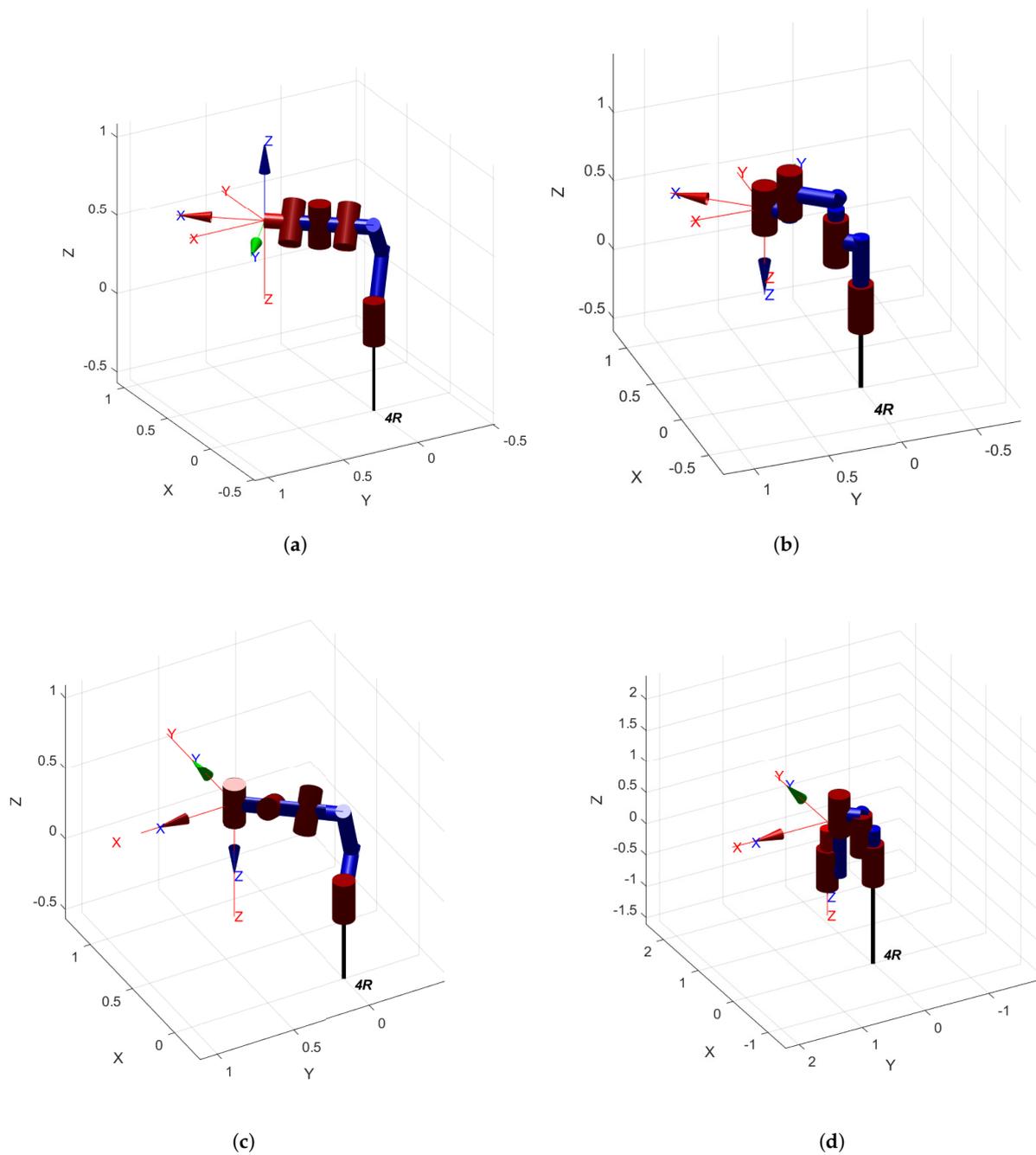


Figure 10. Initial estimation results for Pose (3): (a) type A estimation; (b) type B estimation; (c) type C estimation; and (d) type D estimation.

3.1. Resulting DH Parameters

For a better evaluation, we include Table 1 with the generated DH parameters for *Pose(1)*. The angles θ are considered as variables with zero offset.

Table 1. The DH parameters of manipulators for *Pose*(1).

Joint	d_i (m)	a_i (m)	α_i (rad)	Joint	d_i (m)	a_i (m)	α_i (rad)
1	0.8	0.34	0.79	1	0.64	0.29	-0.39
2	0	0.34	-0.79	2	0.32	0.56	-1.48
3	0	0.34	0.79	3	0.46	0.38	0
4	0	0.34	-0.79	4	0	0	-0.28
a Type A estimation				b Type B estimation			
Joint	d_i (m)	a_i (m)	α_i (rad)	Joint	d_i (m)	a_i (m)	α_i (rad)
1	1.27	0.36	-0.7	1	0.64	0.29	-0.40
2	0	0.36	-0.7	2	0.32	0.55	-1.48
3	0	0.36	-0.7	3	-0.33	0.30	-0.29
4	0.93	0	0	4	0.83	0	0
c Type C estimation				d Type D estimation			

3.2. Manipulator Length Comparison

In general, longer links of a manipulator demand more powerful motors because of higher torques. In Figure 11, there is a bar plot comparing the lengths of the resulting manipulators. The length was determined using Equation (48), where a_i and d_i are DH parameters of i_{th} link. n is the number of joints:

$$L = \sum_{i=1}^n \sqrt{a_i^2 + d_i^2} \tag{48}$$

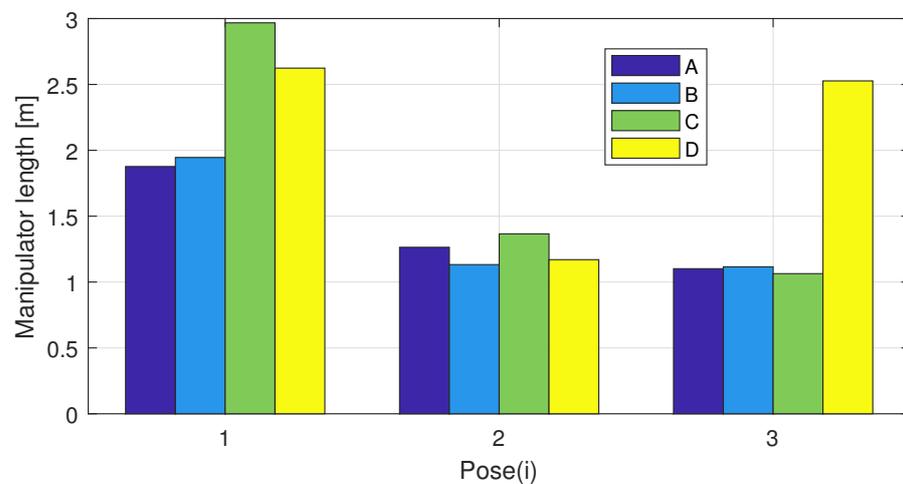


Figure 11. Comparison of the length of generated manipulators for every pose.

3.3. Manipulability Comparison

To compare the results, we decided to evaluate the manipulability of the calculated kinematic structures. This scalar measure was obtained using the Yoshikawa algorithm [1], which describes how spherical the end-effector velocity ellipsoid is. It differs between 0 and 1, where the value 1 shows the best manipulability in all axes. If the value is close to 0, the mechanism might be dealing with singularities.

The results are shown for both translational and rotational motions in Figure 12 as a logarithmic graph, because the measure differs significantly between particular kinematic structures. It should be kept in mind that the presented manipulators have less than six DoFs, which is one of the reasons why the manipulability measure by Yoshikawa evaluates them with low numbers.

As expected, the type A kinematic structure has to deal with singularities and the manipulability tends to be the lowest for the given poses. Type B has almost the same

translational manipulability as type D, but the last two joints are in a singular position, so the rotational manipulability drops in the case of type B kinematic structure. Type C performs better than types A and B. In the case of general poses $Pose(1,2)$, type D provides the highest manipulability measures. However, for $Pose(3)$ when Z axes are parallel, the manoeuvrability of the type D algorithm drops under the values of type C.

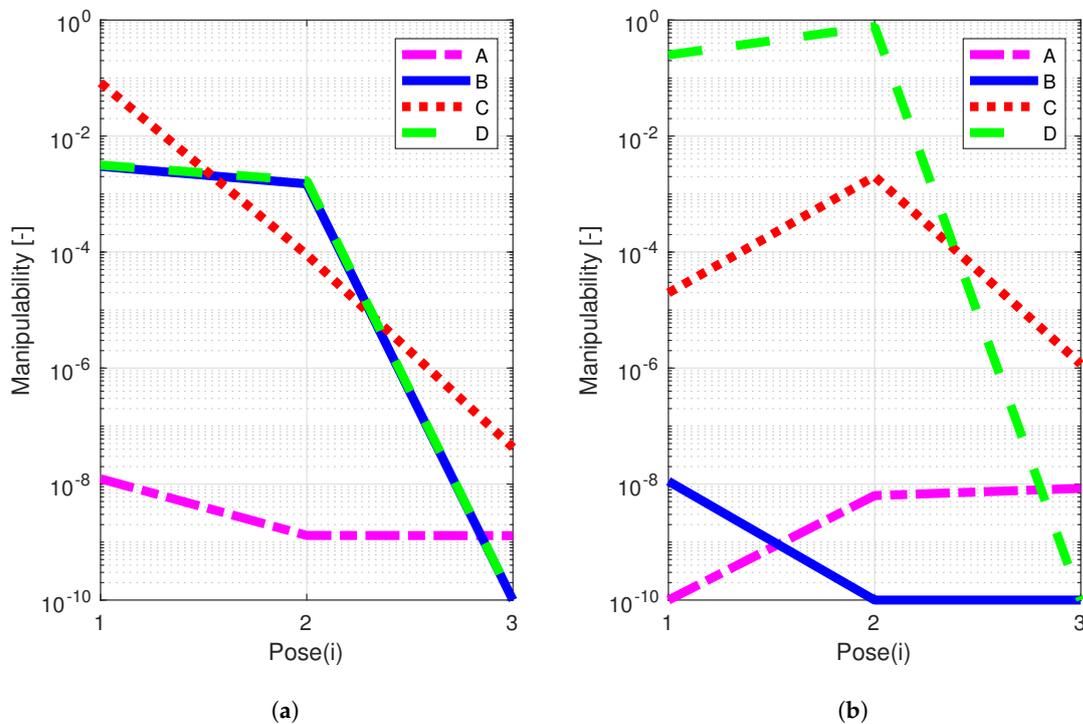


Figure 12. Logarithmic graph comparing the manipulability measures for given poses (1–3): (a) for translational motion; and (b) for rotational motion.

4. Discussion

The outcomes of this paper are aimed to be utilised in the synthesis of kinematic structures of robotic manipulators. Other applications are also possible, for example, in rapid kinematic analysis and simulation to evaluate kinematic options if only one position or pose on a trajectory is desired to be reached. However, only topics related to the synthesis problem are going to be discussed here.

This paper provided four algorithms to estimate the initial value of a kinematic structure for later implementation in optimisation algorithms. As mentioned previously, especially in algorithms searching the local minimum of an objective function, every initial guess can provide different results. Therefore, we decided to present the types (A and B) of estimation even though they do not fulfil the given pose in terms of orientation. The reason is that an optimisation algorithm may overcome this issue later and the solution could achieve the pose on a given trajectory with a better final value of the objective function than with the other presented types (C and D). This always depends on the type and properties of the trajectory. Therefore, we suggest implementing all four estimation types into an optimisation algorithm and compare the results afterwards.

If one would like to know which one of the four structures is the best, this question is not easy to answer. We chose three poses to demonstrate the advantages and disadvantages of particular algorithms and compared the final structures for these poses by manoeuvrability and length. The type D provides the most general structure that can reach any pose, but one must be careful in cases when some axes of the base and the pose are parallel, for example. We suggest to always visualise all initial structures for better evaluation.

The presented algorithms are based on the Denavit–Hartenberg convention, generating DH parameters. During this work, a question arose, of how suitable the DH convention

is for general structures and especially for their synthesis for given trajectory. As mentioned previously, it is possible to find DH parameters between two coordinate frames using the common perpendicular approach, but it is required to “borrow” another two DH parameters from the next joint and to add them to the already existing four parameters. This is obvious since we need six parameters (three translations and three rotations) in general to describe the motion of a frame in Euclidean space. Therefore, using an optimisation algorithm to synthesise a manipulator that fulfils the DH convention may be limiting. The frames representing joints cannot freely rotate and translate wherever the algorithm tends to, but they have to follow the XZ planes in which the common perpendiculars between neighbouring joints lie. On top of that, the algorithm may not find the global or local minimum because of this limit at all. The reason is that traditionally DH parameters serve for the description of existing robots and once they are obtained, some local coordinates of the joints may be located outside of the rigid body along the Z axis, and although they are still closely tied to the particular joint and representing its kinematics, they are not representing the real (physical) position of the joint. On the other hand, in terms of synthesis when a rigid body does not exist yet, the location (transformation) of the coordinate frames of joints is the only known and crucial parameter and its variability should not be limited during the synthesis process anyhow.

The comparison of the synthesis of kinematic structure using DH convention and other standard approaches, such as screw theory [34] for instance, will be an interesting topic for future research. However, this matter has no impact on the work presented in this paper. The kinematic structures obtained using the four algorithms may be translated into any other standard description of the structure of a manipulator.

5. Conclusions

This paper presents four mathematical algorithms to find an initial estimation of a kinematic structure of a serial robotic manipulator. The input values are the number of joints, the position of the base of the manipulator, and the target pose of its TCP. The outputs are the standard DH parameters of a kinematic structure that can reach the given pose either by position or by position and orientation. The presented methods are applicable in the topic of synthesis of the kinematic structure of robotic manipulators, where they can serve as an initial guess of a structure.

The examples of three poses for all four algorithms are shown and compared using the manipulability measure and the arm length. However, it is not easy to determine which method out of these four is the best as it is always depending on the input, especially the TCP pose. The manipulability measure indicates that the D type algorithm, which can fully satisfy any given pose, provides the best manipulability in general, because it avoids placing joints on a single line and places them on a Bézier curve instead. In addition to that, the joints also tend to prevent collisions. On the other hand, if the given pose is representing a specific case, as a parallel axis or axes with a base frame axis or axes, one should be always watchful and comparison with the other presented algorithms is suggested. As expected, algorithm types A and B achieved the best results in the case of the arm length measure, when only a position is desired. However, in some cases, the C and D algorithms may fully achieve a pose with a manipulator with similarly long links.

Future steps are to implement this method in an optimisation algorithm and to observe if the convergence is faster or if the obtained result and the values of objective functions are better.

The algorithms were tested in MATLAB and the scripts are available as Supplementary Material for this paper or with eventual updates as an open source package on a public repository [35].

Supplementary Materials: The source code of algorithms presented in this paper is available as open source repository on the Github page of the Department of Robotics, VSB-Technical University of Ostrava, Czech Republic: github.com/robot-usb-cz/initial-estimation accessed on 15 April 2021.

Author Contributions: Conceptualisation, D.H. (Daniel Huczala) and T.K.; methodology, D.H. (Daniel Huczala); software, D.H. (Daniel Huczala) and P.O.; validation, D.H. (Daniel Huczala), M.P. and D.H. (Dominik Heczko); formal analysis, V.M.; investigation, D.H. (Dominik Heczko); resources, P.O.; data curation, T.K.; writing—original draft preparation, D.H. (Daniel Huczala); writing—review and editing, T.K. and M.P.; visualisation, D.H. (Dominik Heczko); supervision, M.P.; project administration, V.M.; funding acquisition, V.M. All authors have read and agreed to the published version of the manuscript.

Funding: This article has been elaborated under support of the project Research Centre of Advanced Mechatronic Systems, reg. no. CZ.02.1.01/0.0/0.0/16_019/0000867 in the frame of the Operational Program Research, Development and Education. This article has been also supported by specific research project SP2021/47 and financed by the state budget of the Czech Republic.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Source code available on: github.com/robot-vsbs-cz/initial-estimation accessed on 15 April 2021.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CuMa	Curved Manipulator
DoF	Degree of Freedom
DH	Denavit–Hartenberg
GA	Genetic Algorithm
NLP	Nonlinear Programming
SE3	Euclidean Group of Rigid Body Displacements in Three Dimensions
TCP	Tool-Center Point

References

1. Yoshikawa, T. Translational and Rotational Manipulability of Robotic Manipulators. In Proceedings of the 1990 American Control Conference, San Diego, CA, USA, 23–25 May 1990; doi:10.23919/acc.1990.4790733. [CrossRef]
2. Lara-Molina, F.A.; Dumur, D. A Fuzzy Approach for the Kinematic Reliability Assessment of Robotic Manipulators. *Robotica* **2021**, 1–15. [CrossRef]
3. Brandstötter, M.; Angerer, A.; Hofbauer, M. The curved manipulator (cuma-type arm): Realization of a serial manipulator with general structure in modular design. In Proceedings of the 14th IFToMM World Congress, Taipei, Taiwan, 25–30 October 2015; pp. 403–409. [CrossRef]
4. Brandstötter, M.; Gallina, P.; Seriani, S.; Hofbauer, M. Task-Dependent Structural Modifications on Reconfigurable General Serial Manipulators. In *Advances in Service and Industrial Robotics*; Springer International Publishing: New York, NY, USA, 2018; pp. 316–324. [CrossRef]
5. Xu, S.; Li, G.; Song, D.; Sun, L.; Liu, J. Real-time Shape Recognition of a Deformable Link by Using Self-Organizing Map. In Proceedings of the 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), Munich, Germany, 20–24 August 2018; doi:10.1109/coase.2018.8560514. [CrossRef]
6. Clark, A.B.; Rojas, N. Design and Workspace Characterisation of Malleable Robots. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; doi:10.1109/icra40945.2020.9197439. [CrossRef]
7. Pang, Z.; Wang, T.; Wang, Z.; Yu, J.; Sun, Z.; Liu, S. Design and Analysis of a Wearable Upper Limb Rehabilitation Robot with Characteristics of Tension Mechanism. *Appl. Sci.* **2020**, *10*, 2101. [CrossRef]
8. Yan, H.; Wang, H.; Chen, P.; Niu, J.; Ning, Y.; Li, S.; Wang, X. Configuration Design of an Upper Limb Rehabilitation Robot with a Generalized Shoulder Joint. *Appl. Sci.* **2021**, *11*, 2080. [CrossRef]
9. Hauenstein, J.D.; Wampler, C.W.; Pfurner, M. Synthesis of three-revolute spatial chains for body guidance. *Mech. Mach. Theory* **2017**, *110*, 61–72. [CrossRef]
10. Chocron, O.; Bidaud, P. Evolutionary algorithms in kinematic design of robotic systems. In Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems, Innovative Robotics for Real-World Applications, IROS'97, Grenoble, France, 11 September 1997; doi:10.1109/iros.1997.655148. [CrossRef]

11. Chung, W.; Han, J.; Youm, Y.; Kim, S. Task based design of modular robot manipulator using efficient genetic algorithm. In Proceedings of the International Conference on Robotics and Automation, Albuquerque, NM, USA, 25 April 1997; doi:10.1109/robot.1997.620087. [CrossRef]
12. Pastor, R.; Huczala, D.; Bobovský, Z.; Grushko, S. Genetic Optimization of a Manipulator: Comparison between Straight, Rounded, and Curved Mechanism Links. *Appl. Sci.* **2021**, *11*, 2471. [CrossRef]
13. Valsamos, C.; Moulianitis, V.C.; Aspragathos, N. Metamorphic Structure Representation: Designing and Evaluating Anatomies of Metamorphic Manipulators. In *Advances in Reconfigurable Mechanisms and Robots I*; Springer: London, UK, 2012; pp. 3–11. [CrossRef]
14. Katrantzis, E.F.; Aspragathos, N.A.; Valsamos, C.D.; Moulianitis, V.C. Anatomy Optimization and Experimental Verification of a Metamorphic Manipulator. In Proceedings of the 2018 International Conference on Reconfigurable Mechanisms and Robots (ReMAR), Delft, The Netherlands, 20–22 June 2018; doi:10.1109/remar.2018.8449880. [CrossRef]
15. Hamida, I.B.; Laribi, M.A.; Mlika, A.; Romdhane, L.; Zeghloul, S. Dimensional Synthesis and Performance Evaluation of Four Translational Parallel Manipulators. *Robotica* **2020**, *39*, 233–249. [CrossRef]
16. Zeiaee, A.; Soltani-Zarrin, R.; Langari, R.; Tafreshi, R. Kinematic Design Optimization of an Eight Degree-of-Freedom Upper-Limb Exoskeleton. *Robotica* **2019**, *37*, 2073–2086. [CrossRef]
17. Patel, S.; Sobh, T. Task based synthesis of serial manipulators. *J. Adv. Res.* **2015**, *6*, 479–492. [CrossRef] [PubMed]
18. Ha, S.; Coros, S.; Alspach, A.; Bern, J.M.; Kim, J.; Yamane, K. Computational Design of Robotic Devices From High-Level Motion Specifications. *IEEE Trans. Robot.* **2018**, *34*, 1240–1251. [CrossRef]
19. Dogra, A.; Padhee, S.S.; Singla, E. An Optimal Architectural Design for Unconventional Modular Reconfigurable Manipulation System. *J. Mech. Des.* **2020**, 1–29. [CrossRef]
20. Whitman, J.; Choset, H. Task-Specific Manipulator Design and Trajectory Synthesis. *IEEE Robot. Autom. Lett.* **2019**, *4*, 301–308. [CrossRef]
21. Vanderbei, R.J.; Shanno, D.F. An interior-point algorithm for nonconvex nonlinear programming. *Comput. Optim. Appl.* **1999**, *13*, 231–252. [CrossRef]
22. Ghafil, H.N.; Jármai, K. *Optimization for Robot Modelling with MATLAB*; Springer: Berlin/Heidelberg, Germany, 2020.
23. Kot, T.; Bobovský, Z.; Brandstötter, M.; Kryš, V.; Virgala, I.; Novák, P. Finding Optimal Manipulator Arm Shapes to Avoid Collisions in a Static Environment. *Appl. Sci.* **2020**, *11*, 64. [CrossRef]
24. Uicker, J.J.; Denavit, J.; Hartenberg, R.S. An Iterative Method for the Displacement Analysis of Spatial Mechanisms. *J. Appl. Mech.* **1964**, *31*, 309–314. [CrossRef]
25. Barker, L.K. Vector-algebra approach to extract Denavit-Hartenberg parameters of assembled robot arms. In *NTRS-NASA Technical Reports Server*; National Aeronautics and Space Administration, Scientific and Technical Information Branch: Washington, DC, USA, 1983.
26. Corke, P. A Simple and Systematic Approach to Assigning Denavit–Hartenberg Parameters. *IEEE Trans. Robot.* **2007**, *23*, 590–594. [CrossRef]
27. Rajeevlochana, C.; Saha, S.K.; Kumar, S. Automatic extraction of DH parameters of serial manipulators using line geometry. In Proceedings of the 2nd Joint International Conference on Multibody System Dynamics, Stuttgart, Germany, 29 May–1 June 2012; pp. 1–9.
28. Klug, C.; Schmalstieg, D.; Gloor, T.; Arth, C. A Complete Workflow for Automatic Forward Kinematics Model Extraction of Robotic Total Stations Using the Denavit-Hartenberg Convention. *J. Intell. Robot. Syst.* **2018**, *95*, 311–329. [CrossRef]
29. Faria, C.; Vilaca, J.L.; Monteiro, S.; Erlhagen, W.; Bicho, E. Automatic Denavit-Hartenberg Parameter Identification for Serial Manipulators. In Proceedings of the IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society, Lisbon, Portugal, 14–17 October 2019; doi:10.1109/iecon.2019.8927455. [CrossRef]
30. Corke, P. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB®*, 2nd ed.; Springer Tracts in Advanced Robotics; Springer International: New York, NY, USA, 2017.
31. Corke, P. Robotics Toolbox for Matlab. 2020. Available online: <https://github.com/petercorke/robotics-toolbox-matlab> (accessed on 15 April 2021).
32. Brodsky, A. Shortest Distance between Two Lines in N Dimensions. 2011. Available online: <https://www.mathworks.com/matlabcentral/fileexchange/29130-shortest-distance-between-two-lines-in-n-dimensions> (accessed on 15 April 2021).
33. Bai, T. 3D Bezier Curve. 2012. Available online: <https://www.mathworks.com/matlabcentral/fileexchange/35154-3d-bezier-curve> (accessed on 15 April 2021).
34. Lynch, K.M.; Park, F.C. *Modern Robotics*; Cambridge University Press: Cambridge, UK, 2017.
35. Huczala, D. Initial Estimation. 2021. Available online: github.com/robot-vs-bz/initial-estimation (accessed on 15 April 2021).