IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 May 2022, accepted 21 June 2022, date of publication 24 June 2022, date of current version 5 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3186098

RESEARCH ARTICLE

Multirepresentations and Multiconstraints Approach to the Numerical Synthesis of Serial Kinematic Structures of Manipulators

DANIEL HUCZALA[1], TOMÁŠ KOT[1], MARTIN PFURNER[2], VÁCLAV KRYS[1], AND ZDENKO BOBOVSKÝ[1]

[1]Department of Robotics, Faculty of Mechanical Engineering, VSB–Technical University of Ostrava, 70800 Ostrava, Czech Republic
[2]Unit of Geometry and Surveying, University of Innsbruck, 6020 Innsbruck, Austria

Corresponding author: Daniel Huczala (daniel.huczala@vsb.cz)

This work was supported in part by the Project Research Centre of Advanced Mechatronic Systems in the Frame of the Operational Program Research, Development and Education, under Grant CZ.02.1.01/0.0/0.0/16_019/0000867; and in part by the Specific Research Project through the State Budget of the Czech Republic under Grant SP2022/67.

ABSTRACT This paper presents a set of algorithms for the synthesis of kinematic structures of serial manipulators using multiple constraint formulation and provides a performance comparison of different kinematic representations, the Denavit-Hartenberg notation, the Product of Exponentials (screws), and Roll-Pitch-Yaw angles with translation parameters. Synthesis is performed for five given tasks, and both revolute and prismatic joints can be synthesized. Two different non-linear programming optimization algorithms were used to support the findings. The results are compared and discussed. Data show that the choice of the constraint design method has a significant impact on the success rate of optimization convergence. The choice of representation has a lower impact on convergence, but there are differences in the optimization time and the length of the designed manipulators. Furthermore, the best results are obtained when multiple methodologies are used in combination. An arbitrary manipulator was designed and assembled based on a trajectory in the collision environment to demonstrate the advantages of the proposed methodology. The input/output data and synthesis methodology algorithms are provided through an open repository.

INDEX TERMS Robot design, manipulator synthesis, numerical optimization, kinematic representations.

I. INTRODUCTION

The synthesis of a kinematic structure of a robot manipulator is the process of finding kinematic parameters with which the structure fulfills a given task. As a task, one can imagine a path or trajectory that goes through the previously specified poses, i.e., a predefined translation and orientation of the end-effector. In industry, a typical manipulator has the so-called angular kinematic structure with 6 degrees of freedom (DoF), and it can universally serve in various tasks. A synthesized (customized) manipulator can overcome the angular robot in ways such as providing the possibility of avoiding collisions due to its task-specific design, reducing the cost when fewer joints (motors) are needed, or even the ability to change its kinematic structure during the working process as presented by Brandstötter *et al.* [1], where increased temperature of the links makes them flexible, and Clark and Rojas [2], where it is done with the change in air pressure in the links. One can imagine many industrial applications of synthesized manipulators in densely built environments; however, there are also other applications, such as in healthcare to help with upper limb rehabilitation [3], [4], or in maintenance such as arm exoskeletons [5]. If there is a task for which no typical manipulator on the market made by ABB, KUKA, Yaskawa, or others can be deployed, a customized robotic arm can replace them. This study expands the findings of the serial manipulator numerical synthesis problem by introducing multiple representations and multiple constraint methods. The outputs

The associate editor coordinating the review of this manuscript and approving it for publication was Yangmin Li.

can be combined with each other, greatly increasing the number of possible solutions obtained by optimization.

A robot kinematic representation is a mathematical description of the kinematic structure of the robot. Probably the most common mathematical representation of robot kinematics is the Denavit–Hartenberg (DH) convention [6], [7]. Another widely known representation is based on screw theory [8], where the joint axes are represented as normalized twists defined by a directional vector and a linear velocity vector at the origin [9]. It is also called the Product of Exponentials (PoE) representation and is an extension to Plücker coordinates [10]. Lately, due to the application of the Robot Operating System (ROS) and the related Universal Robot Description Format (URDF), the orientation represented by the Roll-Pitch-Yaw (RPY) angles [11] accompanied by displacement along the x, y and z axes has been significantly employed. In this study, we refer to this representation as RPYXYZ.

In numerical optimization, constraints are conditions that must be met at the end of the optimization process. For example, as used in this study, the pose of an end-effector calculated by forward kinematics has to be equal to the pose of a given path.

In this study, we present a robust synthesis methodology that applies fast local optimum search algorithms to obtain the kinematic parameters of a manipulator based on a given task. The algorithms chosen are interior-point (IP) and sequential quadratic programming (SQP). The output provides an arbitrary kinematic structure that does not depend on any predefined set of modules or actuators. The major distinction of the methods already presented is that they always use only one kinematic representation of the robot and only one way of creating constraints. However, the methodology that will be presented applies three well-known and commonly used kinematic representations, DH parameters, RPYXYZ values, and Product of Exponentials (screw) representation, and six methods to create constraints. It will be shown that multiple application of these methods is able to overcome local minimum by starting a new optimization using the previous results that were obtained using different representation or constraints formulation. This can significantly increase convergence and speed of optimization while providing shorter manipulators.

II. RELATED WORK

There are two approaches to solve the synthesis problem of a manipulator. The analytical approach [12] is based on solving algebraic equations to obtain kinematic parameters (expressed algebraically). It mainly focuses on mechanisms interesting from the mathematical point of view, for example, a synthesis of the well-known Bennett mechanism [13]. There are papers dealing with the synthesis of arbitrary mechanisms, as in the work of Hauenstein *et al.* [14] where the synthesis of three-revolute spatial chains for five general poses was solved. If more than 3 degrees of freedom are needed, it becomes very complicated to obtain such a description of a robot analytically. The reason is that the number of possible solutions increases, and the related algebra demands a lot of computational power. Therefore, Perez-Gracia and McCarthy [15] combined their analytical approach with numerical optimization. Today, researchers tend to use optimization algorithms to synthesize manipulators for complex tasks. Mathematical optimization is a set of numerical methods that search for an optimal solution that meets a given objective function while satisfying constraints, if specified. Mostly, global or local minimum search algorithms are applied. Among global optimization algorithms, in the case of synthesis of a robot manipulator, Genetic Algorithms (GA) have been applied in various studies. The comparison of straight, rounded, and curved links was studied by Pastor *et al.* [16], using Schunk actuator modules as joints. The design parameters vary depending on the type of link, without directly specifying any kinematic representation. Valsamos *et al.* and Katrantzis *et al.* [17], [18] also used Schunk actuators with pseudo-joints between them and generated the optimal kinematic structure based on the manipulability of the manipulator. Another global optimum search algorithm is the Simulated Annealing implemented in [19], where they composed a manipulator from predefined links and joints models. Singh *et al.* [20] applied Binary Search Algorithm to synthesize DH parameters to assembly a modular manipulator.

The studies mentioned above were based on the synthesis of predefined motor models, and their approach is limited to those actuators; that is, the output kinematic structures are not purely arbitrary. The synthesis of arbitrary structures was investigated by Patel and Sobh [21] who searched with Simulated Annealing Algorithm the optimal set of Denavit–Hartenberg (DH) parameters for a given goal and then tested the inverse kinematics with Particle Swarm Optimization (PSO). Singla *et al.* [22] obtained DH parameters of an arbitrary manipulator using the Augmented Lagrangian method.

The main disadvantage of global optimum search algorithms is that, for complex tasks, they can search for a solution within minutes, hours, or days, even on today's powerful computers. Therefore, local optimization algorithms can serve better, especially at the beginning of the custom manipulator design process. Dogra *et al.* [23] utilize a non-linear programming method based on the minimization of joint torques, which means that it is a dynamical synthesis and the objective is not to reach given poses, but to fulfill criteria based on energy consumption. The design parameters do not follow any standard kinematic representation, since they include dynamics parameters. Whitman and Choset [24] search for the optimal design of a robotic arm with an objective function that minimizes the length of the path in the joint space. The design parameters are presented vaguely without any specified standardization as a matter of choice and can include ''link lengths, twists about link axes, base locations, or other offsets''. Shirafuji and Ota [25] implemented a synthesis method based on differential inverse

kinematics, where they avoided dual optimization of inverse kinematics and robot design parameters. This contrasts with the majority of applied methods, including the approach presented in this paper. However, one of the disadvantages is that the orientation of the goal pose is not taken into account. The optimization constraints are based on Jacobian, and the outputs are screw coordinates.

Based on our literature review, two major questions are raised and will be addressed in this study. At first, there are multiple robot kinematic representations; however, the cited papers always apply only a single one. The PoE was used in [15], [25] and the DH convention was used in [20]–[22]. The other referenced studies used kinematic parameters defined by themselves, mostly in relation to a given set of actuators or link modules. No studies searched for robot design using the RPYXYZ parameters. The optimization of kinematic structure is a challenging task, and at the same time, all studies tend to distinguish one from each other. Therefore, it is impossible to compare the performance of different representations used in those studies. However, we assume that the choice of representation has an impact on the optimization output.

Second, there are a few ways to define constraints to compare the positioning error between the robot end-effector and the given poses. The goal is to minimize these errors so that a manipulator can reach all given poses with forward kinematics. Again, the presented studies always used only one constraint definition method. In this paper, we compare six methods and - because there is no such limitation - their combinations. To summarize the state of the art in terms of constraints: the norms of the rotational matrix and the translation vector were used in [24], however, the paper does not specify if it is calculating the 2-norm, the Frobenius norm, or some other type of matrix norm. A method that uses the dual-quaternion representation was presented in [15]. RPYXYZ values were applied in [16], [20]–[22]. In addition to that, we added a method that does not apply the norm of the transformation matrix but its elements, a method that uses the norm of the dual quaternions, and we also applied a new method given as a metric in a 12-dimensional Euclidean space E^{12}.

Three kinematic representations and sixty-three combinations of six constraint methods can create up to $3 \cdot 63 = 189$ different local solutions for a single initial estimation (guess of the optimized values), which is often required by a local minimum-based solver. If there are no limits for the initial estimations, there will be a substantial number of possible solutions for a single task. This can serve as an initial population for genetic algorithms [16] or as a data set for the synthesis using neural networks [26]. Other benefits of the presented method are as follows. Synthesis can be performed for both revolute and prismatic joints, which was investigated in the case of arbitrary structures in [21], [22]. The search for the optimal position of a robot base is included in the optimization process. If the output of the synthesis does not satisfy all requirements, it can be used as an initial guess for another iteration using a different representation, constraints, or both.

The methodology is presented in Section III. The results are summarized and discussed in Sections IV and V. We compare the performance of the representation and constraint creation methods for the same input paths and initial estimation. There is no randomness included in the optimization process; that is, the same input provides the same output. In Section VI we present the proof-of-concept of the 3-revolute (3R) mechanism that was built based on a specified collision environment. In this section, one can also find a per-partes optimization process that leads to the final manipulator design. This contrasts with other studies that synthesize kinematics, dynamics, and collision avoidance at the same time. A simple method for avoiding joint collisions with the environment is presented.

III. METHODOLOGY

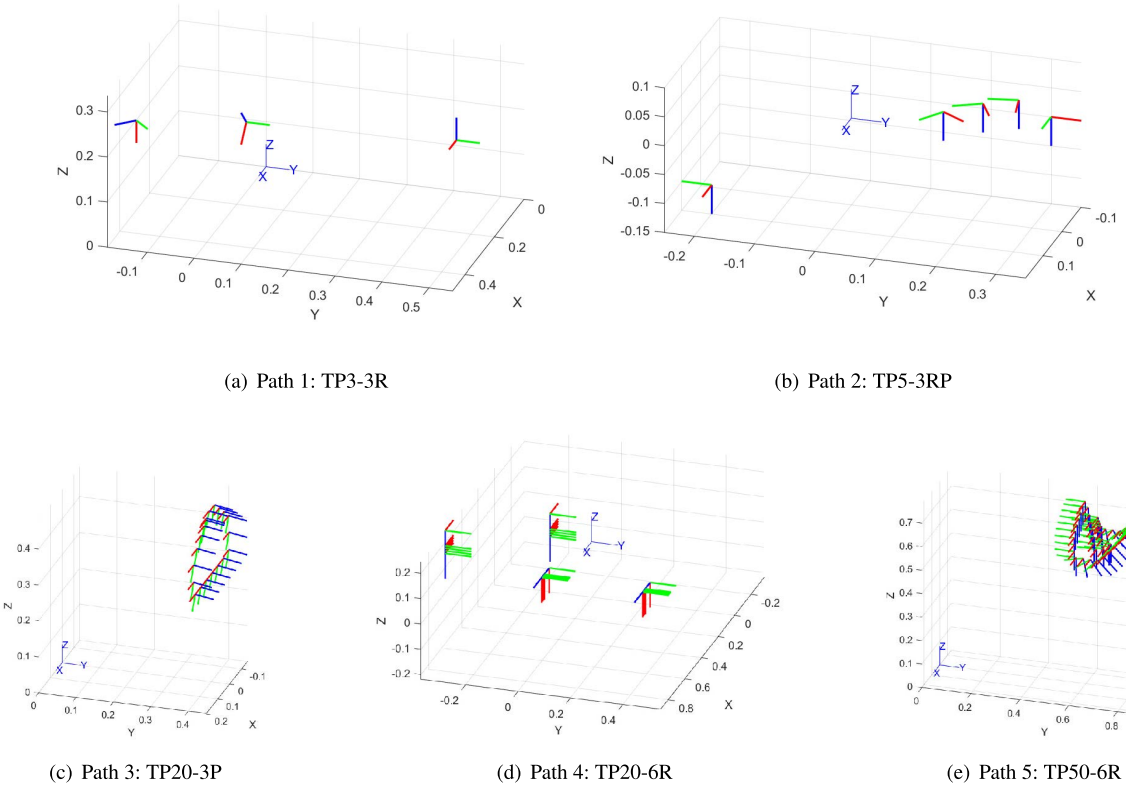
This section is divided into three parts. In the beginning, the mathematical background is briefly introduced. The description of the differences between the optimization of DH, PoE, and RPYXYZ follows. The last part describes the methods for creating constraints.

For optimization itself, the MATLAB^{TM} function *fmincon* [27] was chosen, which is a non-linear programming solver. It searches for the local minimum of an objective function within defined constraints. Beside others, it supports two optimization algorithms that were suitable for the synthesis problem – interior-point (IP) [28] and sequential quadratic programming (SQP) [29]. These algorithms are deterministic (the same input provides the same output) and therefore suit perfectly for measuring the performance of representations and constraints formulation in comparison with, for example, global search algorithms such as PSO or GA, which include some randomization during the optimization process.

In this study, 3780 optimizations of the kinematic structures were performed, as shown in Table 1. The number of representations is three, and the number of constraint types is the combination of six methods, i.e., sixty-three possibilities from using every method individually to using all six combined. Five different paths were chosen; they are shown in Figure 1. For every path, the placement of a base of the robot was restricted in two ways – in the world frame (that is, the position of the base was not optimized) and in the interval $[-0.3, 0.3]$ [m] in every direction around the world frame. Those two options were chosen to demonstrate

TABLE 1. Number of optimizations.

–	Number
Optimization algorithms	2
Representations	3
Constraint types	$\sum_{i=1}^{6} \binom{6}{i} = 63$
Tasks	5
Robot bases	2
Total	$2 \cdot 3 \cdot 63 \cdot 5 \cdot 2 = 3780$



(a) Path 1: TP3-3R (b) Path 2: TP5-3RP

(c) Path 3: TP20-3P (d) Path 4: TP20-6R (e) Path 5: TP50-6R

FIGURE 1. The visualization of the given paths. The captions express number of given task poses (TP), the description after dash line express number and type of joints – revolute (R) or prismatic (P) of the synthesised serial chain; the world frame has blue color.

how the possibility of adjusting the base of a robot during optimization can influence its convergence. One can also see it as a different task (a new given path) when the searched kinematic structure can be completely different.

For each optimization run, the synthesis algorithm requires specifying these inputs by a user:

- Poses of given task,
- interval to place base of the robot,
- number of joints,
- type of joints (revolute or prismatic),
- chosen robot kinematic representation to be synthesized,
- constraint design method.

The output is the numerical parameters of the chosen representation (DH, PoE, RPYXYZ), including the position of its base and end-effector displacement, and the set of joint variables to reach every given pose.

The open-source Matlab toolboxes prepared by Lynch and Park [9] and Corke [30] were used for the calculations presented. The MathWorks Robotics System Toolbox was used for visualization, inspection, and verification.

A. RELATED MATH

In this study, the Special Euclidean group $SE(3)$ is used to represent rigid body displacements as homogeneous transformation matrices. We denote them by $\mathbf{J}$ with axis vectors and position coordinates as shown in (1). $\vec{n}$ (normal) is the X-axis vector, $\vec{o}$ (orientation) is the Y-axis vector, $\vec{a}$ (approach) is the Z-axis vector. They form the rotational matrix **Rot**. $\vec{t}$ (translation) is the position coordinate vector of a pose.

$$\mathbf{J} = \begin{bmatrix} & \mathbf{Rot} & & \vec{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \vec{n} & \vec{o} & \vec{a} & \vec{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Furthermore, the one-to-one mapping of $SE(3)$ in the projective space $\mathbb{P}(\mathbb{R})^7$ was used over the eight-dimensional vector space of dual quaternions [31]. Dual quaternion p

$$p = p_0 + \epsilon p_1 \tag{2}$$

consists of

$$p_0 = a_0 + a_1\vec{i} + a_2\vec{j} + a_3\vec{k} \tag{3}$$

which is the quaternion representing orientation, and

$$p_1 = c_0 + c_1\vec{i} + c_2\vec{j} + c_3\vec{k} \tag{4}$$

which is the dual part representing the position in space. Together, they form homogeneous coordinates $(a_0 : a_1 : a_2 : a_3 : c_0 : c_1 : c_2 : c_3)$ in $\mathbb{P}(\mathbb{R})^7$, where the elements of $SE(3)$ can be represented by points. One of the benefits of this representation is that both position and orientation can be expressed in a single vector.

Joints can be kinematically expressed using homogeneous transforms, or as normalized twists called the screw axis $\mathcal{S}$, which is 6×1 vector

$$\mathcal{S} = \begin{bmatrix} \vec{\omega} \\ \vec{v} \end{bmatrix} \tag{5}$$

The vector $\vec{\omega}$ is the direction of the rotational axis and corresponds to the vector $\vec{a}$ of a transformation matrix. A coordinate q is any point on the axis. Their cross product $\vec{v} = -\vec{\omega} \times q$ forms a moment of the axis about origin. Note the difference with the Plücker coordinates where $\vec{v'} = \vec{\omega} \times q$. The forward kinematics of a manipulator is calculated using the product of exponentials formula, as shown in (6). The matrix $\mathbf{M}$ is the transformation from a manipulator base to its end-effector in the home configuration. $[\mathcal{S}_i]$ is the representation of a screw axis written in matrix (skew-symmetric) form. θ stands for the set of joint variables.

$$\mathbf{J}(\theta) = e^{[\mathcal{S}_1]\theta_1} \dots e^{[\mathcal{S}_n]\theta_n}\mathbf{M} \tag{6}$$

On the other hand, the forward kinematics of a manipulator represented by DH or RPYXYZ parameters is calculated as the multiplication of partial transformation matrices between the joints.

$$\mathbf{J}(\theta) = \mathbf{A}_{01}\mathbf{A}_{12}(\theta_1) \dots \mathbf{A}_{i-1,n}(\theta_n) \tag{7}$$

where, in the case of DH convention,

$$\mathbf{A}_{i-1,i} = \mathbf{R}_{z_{i-1}}(\theta_i)\mathbf{T}_{z_{i-1}}(d_i)\mathbf{T}_x(a_i)\mathbf{R}_x(\alpha_i) \tag{8}$$

is obtained by multiplying the rotation matrix $\mathbf{R}_z(\theta_i)$ around the z_{i-1} axis, translation matrix $\mathbf{T}_z(d_i)$ along the z_{i-1} axis, translation matrix $\mathbf{T}_x(a_i)$ along the x_i axis, and rotation matrix $\mathbf{R}_x(\alpha_i)$ around the x_i axis. Clearly, the translation along and rotation around the y axis is missing – this is overcome by placing the coordinate frames in a special order following the Denavit-Hartenberg convention.

The displacement calculated with RPYXYZ parameters is

$$\mathbf{A}_{i-1,i} = \mathbf{T}_i(x_i, y_i, z_i)\mathbf{R}_{z_i}(\gamma_i)\mathbf{R}_y(\beta_i)\mathbf{R}_x(\alpha_i) \tag{9}$$

where $\mathbf{T}_i$ is the translation matrix between the origins of the two frames and $\mathbf{R}_{x_i}, \mathbf{R}_{y_i}, \mathbf{R}_{z_i}$ are rotations around the axes with angles α, β, γ representing the Roll-Pitch-Yaw angles, respectively.

In the next sections, the norms of vectors and matrices are used. They are all calculated as the 2-norm (Eucleadean length) type.

B. REPRESENTATIONS IN THE SYNTHESIS ALGORITHM

The presented algorithm design differs in some parts for the three considered representations; however, the purpose of this study was to keep as much as possible in common for all those three variants to provide a relevant comparison.

The objective function is to minimize the final length of the manipulator, i.e., reducing the torques needed in actuators while meeting the given constraints and optimized parameter boundaries. At the beginning of every iteration, the algorithm checks whether the forward kinematics of the robot is equal to the given poses. If there is a positioning error (constraints are violated), the algorithm adjusts the optimized parameters, i.e., the robot kinematic structure, and passes them to the next iteration to observe how the objective function and constraints violation values have changed. It is important to note that, along with the optimization of the kinematic parameters, the algorithm solves the inverse kinematics problem at the same time.

In the options of the *fmincon* solver, the maximum iteration value was increased to 300. The number of maximum function evaluations was set to 30, 000. The Step Tolerance was set to 10^{-300}. The other function options were kept as default values. The *fmincon* requires an initial estimation of the optimized parameters. For a local optimum search, this first guess may have a major impact on the outcome. To avoid any randomness, we applied the *methodology D* described in our previous paper [32], where four methods are presented to create kinematic structures represented in the DH parameters using geometrical analysis between a given pose and the base of the manipulator. The methodology D places the joints according to the DH convention on a Bézier curve. The last joint is placed out of the curve but in a way that the forward kinematics of the serial chain can reach the chosen pose, in this study, the most distant pose on the path. The distance is measured from the center of the given interval where the base of the robot can be placed. The *fmincon* solver also requires the specification of the bounds. These limits were set as presented in Table 2. The length and moment parameters are determined again in relation to the most distant pose of a given path and the norm of its position vector $\vec{t}$. The dual quaternions are in projective space; therefore, the bound is set to infinity.

TABLE 2. Variable boundaries.

Kinematic parameter type	Boundary
Angle value	$\pm\pi$ [rad]
Length value	$\pm\lvert\vec{t}\rvert/2$ [m]
Dual quaternion element	$\pm\infty$ [-]
Screw direction element ($\omega_{1..3}$)	± 1 [-]
Screw moment element ($v_{1..3}$)	$\pm 3\lvert\vec{t}\rvert/2$ [-]

The base of a robot is represented as a dual quaternion p_b with the orientation part set to the identity ($p_{b_0} = 1; a_0 = 1, a_1 = 0, a_2 = 0, a_3 = 0$), that is, no rotation is allowed, and only the dual part p_{b_1} (base translation) is being optimized. The end-effector is represented as dual quaternion p_e without any optimization restrictions. These are in total the first twelve optimization parameters that enter the optimization process:

$$n_{be} = n_b + n_e = 4 + 8 = 12 \tag{10}$$

Note that if no base interval is given, then the robot base frame displacement is not optimized, but set to the input value or the world frame.

The following subsections describe how the algorithm design varies according to the chosen robot kinematic representation.

1) DENAVIT–HARTENBERG PARAMETERS

The objective function is to minimize the length parameters given by the DH convention, that is, the absolute distances a_i and d_i between the axes z_{i-1} to z_i and x_{i-1} to x_i of the joints, as shown in (11).

$$f_{DH}(x) = ||\vec{t}_e|| + \sum_{i=1}^{n_j+1} |a_i| + \sum_{i=1}^{n_j+1} |d_i| \tag{11}$$

where n_j is the number of joints and $\vec{t}_e$ is the 4×1 vector of the dual part (position) of the end-effector displacement – the partial transform from the last joint. In each iteration of the optimization process, the forward kinematics of the current manipulator $\mathbf{J}_{1..n}(\theta_{i,n})$ is calculated and compared to the target pose values $\mathbf{P}_{1..n}$. The number of poses given is n. The difference (positioning error) is processed to define the constraints using one to six methods. The details are given in Section III-C.

Equation (11) works with scalar norms. We also tried to implement its version using squares of scalars, for example a_i^2 instead of $|a_i|$, which was expected to provide a better output during optimization since the square function is continuous and differentiable; however, the results were worse than compared to the objective function presented. This is probably due to internal calculations of the *fmincon* solver that can better handle such input.

2) PRODUCT OF EXPONENTIALS

The objective function for the screw representation is designed as follows.

$$f_{PoE}(x) = ||\vec{m}|| + \sum_{i=1}^{n_j} ||\vec{v}_i|| \tag{12}$$

where $\vec{m}$ is 4×1 vector of the dual part (position) of the end-effector displacement – the distance of the matrix $\mathbf{M}$ from the base frame. The other element minimized is the sum of moments of the screws $\vec{v}_i$.

Constraints are created in a similar way as in the case of synthesis using DH parameters. The forward kinematics of the current manipulator $\mathbf{J}_n(\theta_{i,n})$ is calculated using the product of exponentials formula (6). The results are compared with the goal poses $\mathbf{P}_{1..n}$.

The DH parameters a_i, d_i, and α_i have no relation to each other, that is, the value of one element has no impact on the value of the others. That is not the case for screws, where two conditions apply. First, as already mentioned, the screws have to be normalized – the part representing the direction of every joint axis $\vec{\omega}_i$ must obey

$$||\vec{\omega}_i|| = 1 \tag{13}$$

In addition to that, every pair of vectors $\vec{\omega}_i$ and $\vec{v}_i$ preserves the perpendicularity between these vectors and their dot product shall be equal to zero.

$$\vec{\omega}_i \cdot \vec{v}_i = 0 \tag{14}$$

Therefore, equations (13) and (14) are also added to the set of optimization constraints.

3) ROLL-PITCH-YAW ANGLES AND XYZ TRANSLATION VECTOR

The objective function for the RPYXYZ parameters is set

$$f_{rpyxyz}(x) = ||\vec{t}_e|| + \sum_{i=1}^{n_j} ||\vec{t}_i|| \tag{15}$$

where $\vec{t}_e$ is again the dual part of the end-effector displacement in relation to the last joint frame, $\vec{t}_i$ is the vector of the translational coordinates XYZ between consecutive joints.

C. CONSTRAINTS

Only the type of equality constraints of the *fmincon* solver are applied in the presented methodology, and the inequality constraint vector remains empty. The algorithm tries to minimize the objective function (provide as short linkages as possible) which is expected to not be zero in the end, but on the other hand, it has to satisfy the given constraints, i.e., the constraint violation is not allowed and the sum of constraint errors has to be minimized to zero.

Let us define the sum of errors E – the constraint violation value as the sum of all constraints equality equations ceq.

$$E = \sum_{i=1}^{k} ceq_i \sim \sum_{i=1}^{k} (\mathbf{J}_{n,k}(\theta_{n,k}) - \mathbf{P}_{n,k}) \to 0 \tag{16}$$

which is equivalent to the difference between the forward kinematics of the synthesized arm $\mathbf{J}_{n,k}(\theta_{n,k})$ and the given poses $\mathbf{P}_{n,k}$. The number of constraining equations k depends on the chosen method of constraint creation and, because they can be combined, also on their combination.

For each given pose $P_{1..n}$, the constraints are calculated in every iteration and based on this output, the kinematic structure design parameters are altered and parsed in the next iteration.

1) METHOD A

There is a set ceq_A of twelve constraint equations defined as

$$ceq_{1..9} = w(R_{i,\mathbf{J}(\theta)} - R_{i,\mathbf{P}}) \tag{17}$$

$$ceq_{10..12} = \vec{t}_{\mathbf{J}(\theta)} - \vec{t}_{\mathbf{P}} \tag{18}$$

$$w = 0.1 \tag{19}$$

where R_i represents $i = 1..9^{\text{th}}$ element of rotational part of matrix $\mathbf{J}$ or $\mathbf{P}$, $\vec{t}$ represents x, y, z elements of the translational vector of homogeneous matrices $\mathbf{J}$ or $\mathbf{P}$. w is the weight that compensates the ratio between meters and radians.

2) METHOD B

There is a set ceq_B of six constraint equations defined as

$$ceq_1 = w(\alpha_{\mathbf{R_{J(\theta)}}} - \alpha_{\mathbf{R_P}}) \quad (20)$$
$$ceq_2 = w(\beta_{\mathbf{R_{J(\theta)}}} - \beta_{\mathbf{R_P}}) \quad (21)$$
$$ceq_3 = w(\gamma_{\mathbf{R_{J(\theta)}}} - \gamma_{\mathbf{R_P}}) \quad (22)$$
$$ceq_{4..6} = \vec{t}_{\mathbf{J}(\theta)} - \vec{t}_{\mathbf{P}} \quad (23)$$

where α, β, γ are roll-pitch-yaw angles obtained from rotational matrix $\mathbf{R}$ of $\mathbf{J}$ or $\mathbf{P}$. This method was used in [16], [20]–[22]. Weight $w = 0.1$ again.

3) METHOD C

There is a set ceq_C of four constraint equations defined as

$$ceq_1 = w||\vec{n}_{\mathbf{R_{J(\theta)}}} - \vec{n}_{\mathbf{R_P}}|| \quad (24)$$
$$ceq_2 = w||\vec{o}_{\mathbf{R_{J(\theta)}}} - \vec{o}_{\mathbf{R_P}}|| \quad (25)$$
$$ceq_3 = w||\vec{a}_{\mathbf{R_{J(\theta)}}} - \vec{a}_{\mathbf{R_P}}|| \quad (26)$$
$$ceq_4 = 6||\vec{t}_{\mathbf{J}(\theta)} - \vec{t}_{\mathbf{P}}|| \quad (27)$$

where the vectors $\vec{n}, \vec{o}, \vec{a}$ represent x, y, z axis vectors of the two poses. This metric is presented in [33] using a 12-dimensional Euclidean space E^{12}. Weight $w = 0.1$.

4) METHOD D

There is a set ceq_D of two constraint equations defined as

$$ceq_1 = w||\mathbf{R}_{\mathbf{J}(\theta)} - \mathbf{R}_{\mathbf{P}}|| \quad (28)$$
$$ceq_2 = ||\vec{t}_{\mathbf{J}(\theta)} - \vec{t}_{\mathbf{P}}|| \quad (29)$$

where the 2-norms of $\mathbf{R}$ and $\vec{t}$ are taken into account instead of their elements. This method was used in [24]. Weight $w = 0.1$.

5) METHOD E

There is a set ceq_E of eight constraint equations using elements of dual quaternions

$$ceq_{1..8} = p_{\mathbf{J}(\theta)} - p_{\mathbf{P}} \quad (30)$$

where $p_{\mathbf{J}(\theta)}$ and $p_{\mathbf{P}}$ are dual quaternion representation of $\mathbf{J}$ and $\mathbf{P}$. Similar method was used in [15].

6) METHOD F

There is only one ceq_E constraint equation defined as

$$ceq_1 = ||p_{\mathbf{J}(\theta)} - p_{\mathbf{P}}|| \quad (31)$$

which is the norm between those two dual quaternions.

IV. RESULTS

This section presents an analysis of the output data from the optimization process. The input data (paths), multi-representations and multi-constraints synthesis algorithm, and the results are available on public repository [34]. Based on the analysis of obtained data, the performance of the representations is similar; however, the choice of constraint design seems to have a major impact on the overall synthesis problem. An example of an output kinematic structure from a single optimization run out of those 1890 is visualized in Figure 2.

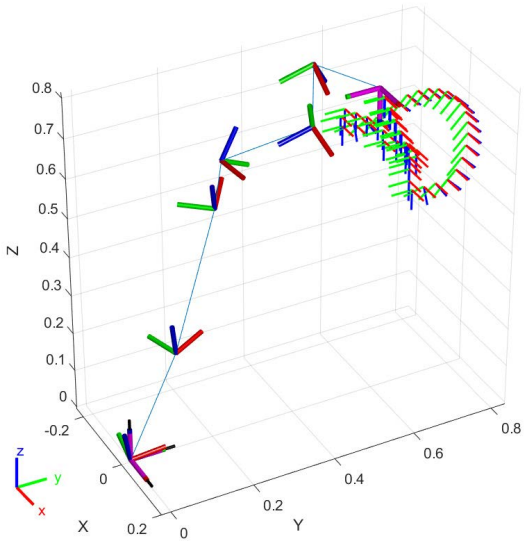


FIGURE 2. Visualization of an optimized kinematic structure performing Path 5; the base and the end-effector frames are pink; the joint frames have RGB color, joint rotational axis is blue.

The convergence success rate is presented. It shows how often the algorithm provides an output with the constraint violation value $E < 10^{-5}$, i.e. the structure is able to precisely reach all poses on a given path; see again Equation (16) for details. Another parameter that is compared is the length of the optimized manipulator, calculated as the sum of the distances between consecutive joint axes given by the nearest points on those axes. Instead of the number of iterations, which did not differ much for the same inputs, the optimization time is provided. The length of the manipulator and the optimization time are also presented only for the outputs with $E < 10^{-5}$.

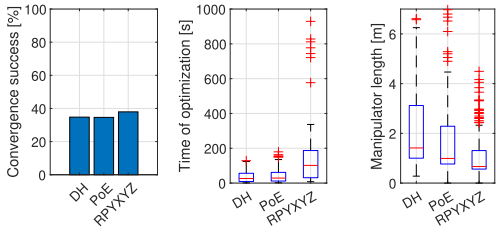
A. OPTIMIZATION WITH INTERIOR-POINT ALGORITHM

Table 3 shows how the complexity of a given path can influence the output, i.e. when the given task is more complicated from the synthesis point of view, the convergence success rate drops.

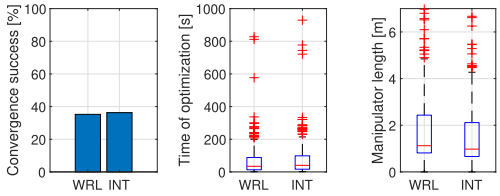
TABLE 3. Convergence success rate, 378 runs performed per path.

Path number	DH	PoE	RPYXYZ	Average
1	72.2 %	85.7 %	79.4 %	79.1 %
2	12 %	9.5 %	12.7 %	11.4 %
3	81.1 %	67.2 %	85.0 %	77.8 %
4	3.1 %	4.0 %	6.0 %	4.5 %
5	5.6 %	6.4 %	5.6 %	5.8 %

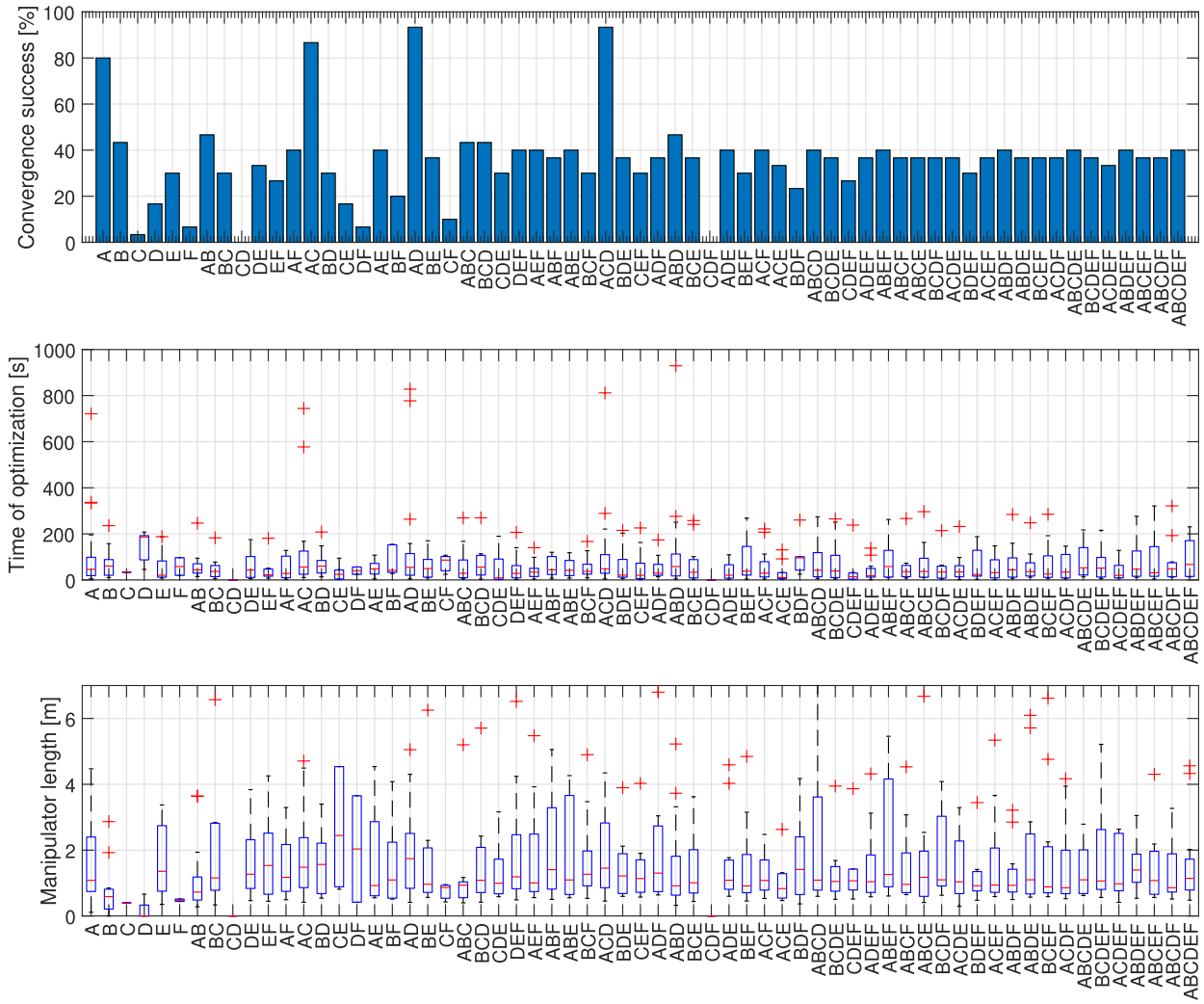
In Figure 3(a), the performance of robot kinematic representations can be observed. In general, there are no unexpected differences between them. For every representation, 630 optimizations were performed. As already mentioned



(a) Performance of robot kinematic representations; convergence success rate, optimization time, manipulator length.



(b) Impact of base specification; convergence success rate, optimization time, manipulator length; WRL – base of the robot was locked in the world frame, INT – position of the base was given by interval and part of the optimization.



(c) Performance of constraint combination methods for all paths; convergence success rate, optimization time, manipulator length.

FIGURE 3. Analysis for IP algorithm – comparison of representations, base specification, and constraint design methods.

in the Introduction part, the RPYXYZ was not used in any research, but based on the results, it provides the highest convergence success rate and the shortest linkages, even though it tends to take more time to provide the output. The reason is probably the number of parameters that are optimized. In addition to the twelve parameters of the base and end-effector frames (shown in Equation (10)) that all three representations share, the number of parameters for each joint differs. For DH parameters, it is a, d, α, and the joint variable θ, that is, four parameters per joint. The PoE

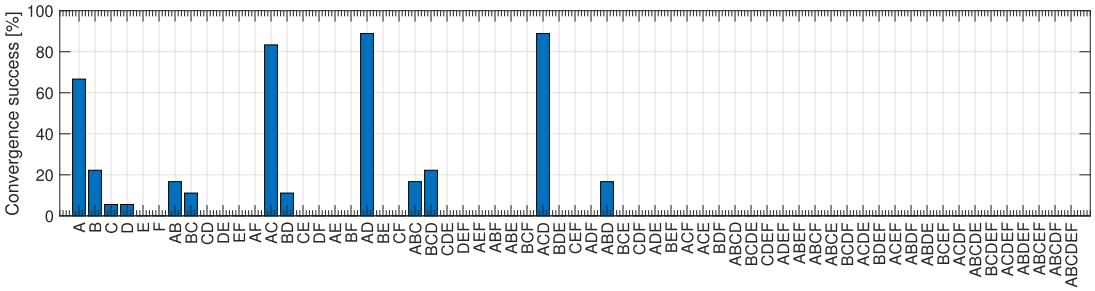


FIGURE 4. Performance of constraint combination methods only for complex Paths 2, 4, and 5 – convergence success rate.

representation has 6×1 screw vector $\mathcal{S}$ plus joint variable θ, but two constraints must be neglected as explained in equations (13) and (14). That is, five parameters per joint. The RPYXYZ representation has five parameters plus the yaw angle that is considered as a joint variable, i.e. six parameters per joint in total. In a 3D space, there are 6 degrees of freedom; therefore, the RPYXYZ representation seems to be the most relevant. However, the "minimal" approach of DH parameters brings advantages, for example as examined, faster optimization time. In the case of numerical optimization of spatial chains, these results indicate that researchers relying on a single representation may be losing possible configurations of their kinematic structures.

Figure 3(b) shows the comparison between the bases of the robot given directly and by an interval. For both, 945 optimization runs were done. One can see that the interval option converges more often and provides shorter links; however, the difference is not significant. On the other hand, the interval may play a key role if a given trajectory itself requires fewer degrees of freedom. An obvious example is Path 3, where, if the interval of the Y-axis was covering the planar part of the trajectory, only two prismatic DoFs would be necessary.

The output data in Figure 3(c) focusing on constraint design methods show that this choice has an impact on the optimization results. For each method or combination of methods, 30 optimization runs were performed. The exceptional outcome is provided by method A and its combination with methods C and D. On the other hand, method C itself has a very poor performance. Method B, which was applied most often in other studies of robot synthesis, has a convergence success rate 43%, and for comparison, the combination method AD has a success rate of 93%. Unfortunately, no other paper shares the convergence success rates of their algorithms. Some of them state that the convergence success rate is not always certain in their case [15], [16], [22]. The lack of such measures is even more visible in Figure 4, where the convergence success rate is shown only for complicated (from the synthesis point of view) Paths 2, 4, and 5. Although some methods keep their success rate similar regardless of what task is given, many other methods do not achieve even a single successful run.

Another interesting observation is that the combination of constraint methods neither extends the optimization time nor seems to have an impact on the length of the manipulator. The shorter optimization time indicates that the evaluation of constraint violation is fast enough in comparison with other evaluation parts of the algorithm. However, based on the data obtained, the combination of more than three methods does not produce a higher success rate.

B. OPTIMIZATION WITH SQP ALGORITHM

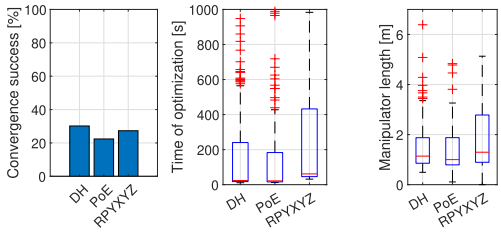
In Figure 5, the same data are provided for the SQP algorithm. The average manipulator length is similar, the convergence success rate is slightly lowered, and the optimization time is extended a lot. Therefore, the IP algorithm provides better results than SQP.

The most interesting comparison provides the plot of the convergence success rate for the constraint design methods. The performance of particular methods A to F is reduced. The most successful combinations of AC, AD and ACD also decreased, but not so much. This leads to the suggestion that the combination of AC, AD, or ACD methods should be used as the first choice in future research on the problem of synthesis of kinematic structures.

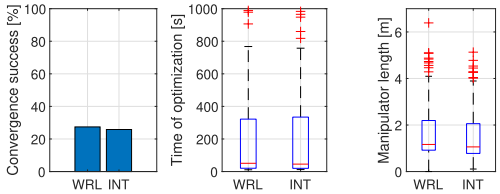
V. DISCUSSION

The serial manipulator synthesis problem using numerical optimization is a very challenging and time-consuming task, where many variables can have an impact on the outcome. Therefore, we do not want to claim that constraint design method A and its combination are the best choice for the synthesis of robot kinematic structures. It can be a coincidence that these methods suffer less in terms of singularities in combination with the chosen optimization solver *fmincon* and the two chosen algorithms. This shall be verified in future studies; however, we want to highlight this problem because the constraints design methodology has not been studied before and seems to be of particular importance. Moreover, the implementation of other constraint methods is not as challenging as the implementation of another robot representation; it can be done and checked for convergence relatively quickly.

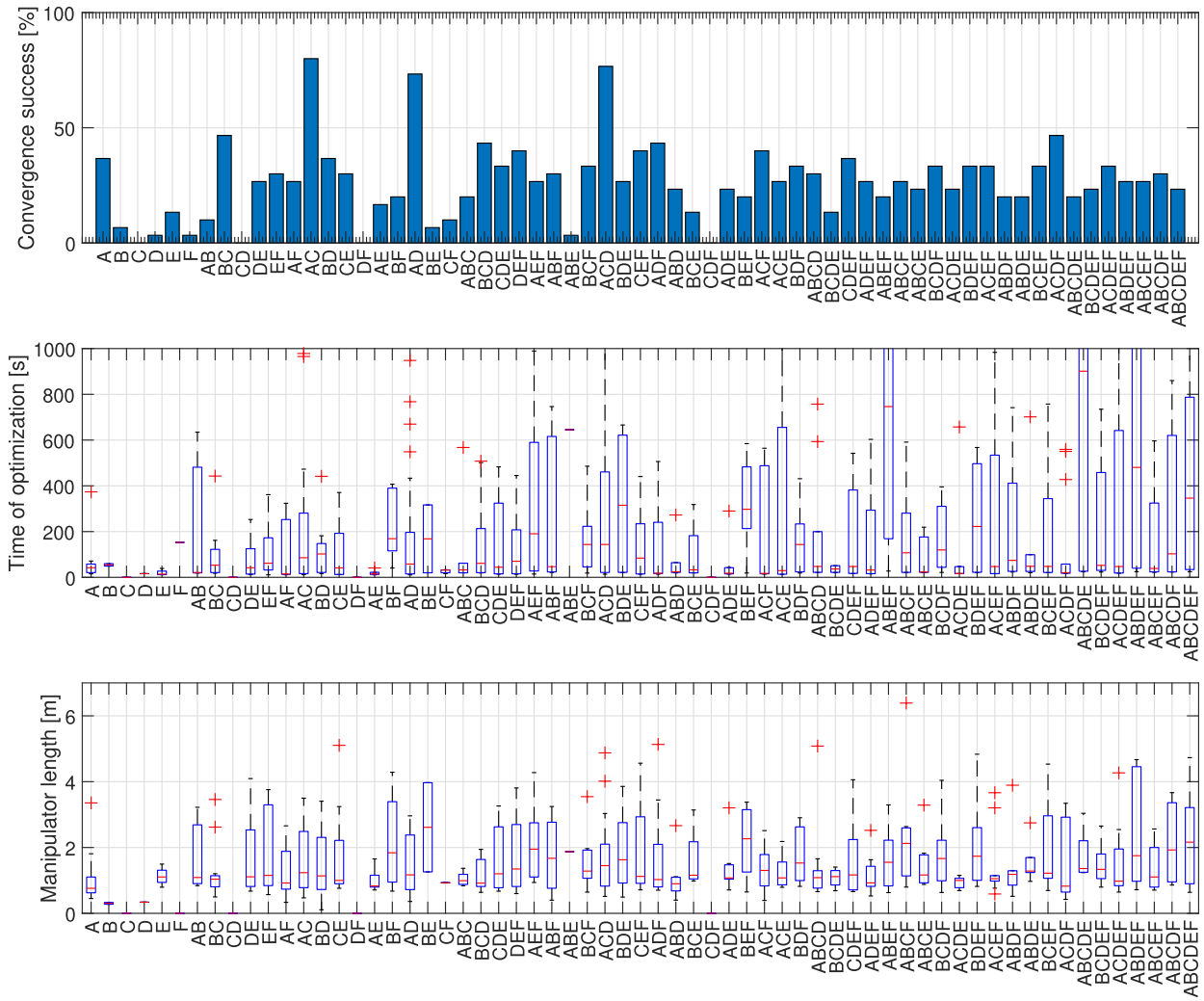
On the other hand, all three investigated representations perform in a similar way, and their selection is not crucial for the output. It is necessary to mention that the DH parameters were struggling with their convergence until a tool frame was



(a) Performance of robot kinematic representations; convergence success rate, optimization time, manipulator length.



(b) Impact of base specification; convergence success rate, optimization time, manipulator length; WRL – base of the robot was locked in the world frame, INT – position of the base was given by interval and part of the optimization.

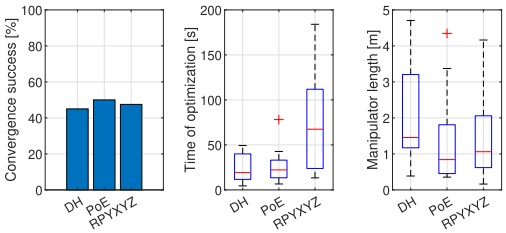


(c) Performance of constraint combination methods for all paths; convergence success rate, optimization time, manipulator length.

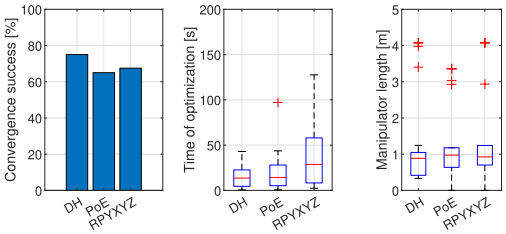
FIGURE 5. Analysis for SQP algorithm – comparison of representations, base specification, and constraint design methods.

added and its displacement optimized along with the joint parameters.

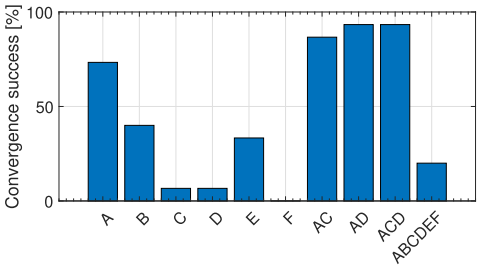
The presented results led us to the idea of running only one optimization for every path using the successful method A in combination with the DH parameters and applying those results as the initial estimation for these ten other constraint design methods: A, B, C, D, E, F, AC, AD, ACD, ABCDEF, and using all three representations. The mapping between the representations was achieved using our algorithms presented in [35]. The base frame position was fixed in the world frame,



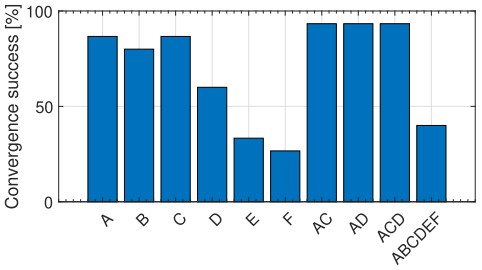
(a) Representations; default initial estimation.



(b) DH parameters and constraint method A as initial estimation.



(c) Constraints; default initial estimation.



(d) DH parameters and constraint method A as initial estimation.

FIGURE 6. **Comparison of performance of constraint methods A, B, C, D, E, F, AC, AD, ACD, and ABCDEF for every path.**

it was not optimized. The IP algorithm was chosen as it was shown that it converges better and provides results faster.

Figure 6 shows that applying the initial estimation that can reach every given pose already at the beginning of the optimization process significantly decreases both the optimization time and the length of the manipulator and increases the convergence success rate. Note that it was still achieved by avoiding any randomness in the optimization process. This fast iterative procedure may speed up the overall time of the synthesis and custom manipulator design process. While the DH convention and the RPYXYZ parameters share the idea of partial transformations between joints,the PoE uses a completely different computational approach for direct and inverse kinematics problems, so switching between these representations during synthesis, especially in local minima search algorithms, can help overcome a local minimum to get closer to the global minimum. The workflow of such an optimization approach is visualized in Figure 7.

There are other variables that need to be investigated in more detail. We chose the weight $w = 0.1$ to compensate the relation between meters and radians based on our experience with overall performance. Definitely, every constraint method would perform differently if an exact value was determined for each separately on the basis of its performance. However, the value of w can also vary based on the number of poses of a given path and the overall size of the trajectory (and output manipulator). This requires another broad study. On the other hand, even though in some presented methods the translantion may seem preferred over the orientation with this weighing, we strived to reduce its impact using a very strict convergence threshold (sum of positioning errors) $E < 10^{-5}$.

So far, six constraint design methods have been examined. These can be extended to other ones, e.g. the combination of a

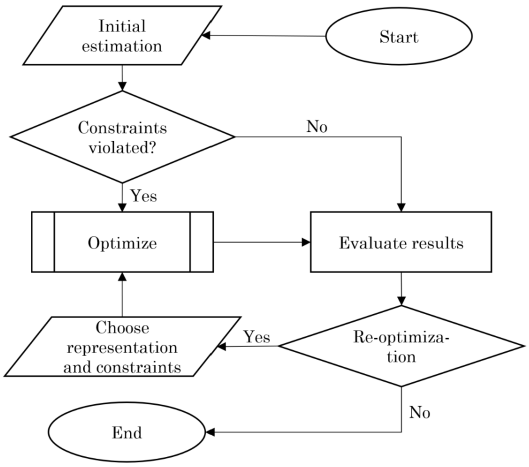


FIGURE 7. **Re-optimization process to avoid local minima.**

quaternion for rotation and XYZ coordinates for translation, their norms, or even other metrics such as Method C. Also there is a possibility to test performance of other norm types than 2-norm, for example the Frobenius norm.

VI. PROOF OF CONCEPT

This chapter describes how the presented methodology can be applied in the design of serial robotic manipulators. After the optimal kinematic structure is synthesized, there are a few challenges related to dynamics and collision avoidance problems. In contrast to other studies [16], [19], [20], [22]–[24] that try to synthesize kinematics along with dynamics and/or collision avoidance at the same time,

we propose a per-partes optimization approach. This means dividing the optimization process into steps:

1) Early synthesis of kinematic structure to obtain better initial estimation of the optimized values.
2) Synthesis of kinematic structure using the output from the previous optimization as initial estimation, changing representations and/or constraint design methods to achieve optimal result.
3) Finding collision-free position of joints.
4) Finding collision-free shapes of links.
5) Choosing the motors and link design based on dynamic analysis.

If the optimization process takes into account many parameters (variables), it becomes time demanding and the convergence is less feasible. The proposed division may decrease the time needed for optimization with the possibility of finding results that could be difficult to achieve if the procedure is tried by brute-force optimization of all possible variables. In this paper, we have investigated so far Steps 1) and 2). Note that the optimization process may involve an extended objective function to take into account joint velocities [25], torques [24], or other constraints. However, this is not necessary.

For Step 3), a simple solution is proposed in Subsection VI-A. Optimization step 4) was addressed in [36], where the links between the joints are shaped using Bézier curves to avoid collision on a trajectory. Step 5) has been investigated in other studies, for example, in the case of links [37] and in the case of choosing the right actuators [38].

A. FINDING COLLISION-FREE POSITION OF JOINTS

The *fmincon* function was implemented again with a similar goal as in the previous ones, the objective being to keep the links as short as possible:

$$f_c(x) = \sum_{i=3}^{n_j+1} ||\vec{t}_i|| \tag{32}$$

where $||\vec{t}_i||$ is the distance between two consecutive joints in RPYXYZ representation. The start index is $i = 3$, which is the displacement between the first and second joints in this representation. The displacement between the world frame and the robot base frame has $i = 1$, the displacement between the base frame and the first joints with $i = 2$ is expected to be in a position without any collision.

The physical position of every joint i represented by a homogeneous matrix $\mathbf{J_i}$ can be expressed as a point r_i on its rotation axis using the parametric equation of a line:

$$ceq_{i-1,i=2..i} = r_i = \vec{t}_i + s_i\vec{a}_i \tag{33}$$

with parameter s_i. The vector $\vec{t}$ represents the position and $\vec{a}$ the vector of the z axis of $\mathbf{J_i}$. Equation (33) therefore forms a set of constraint equations. Another constraint set deals with collisions. The environment is represented by spheres of diameter d_s and the joints by spheres of diameter d_j.

$$ceq_k = \begin{cases} 0, & \text{if } ||\vec{t}_j - \vec{t}_s|| > \dfrac{d_j + d_s}{2} \\ \dfrac{d_j + d_s}{2} - ||\vec{t}_j - \vec{t}_s||, & \text{otherwise} \end{cases} \tag{34}$$

where k is the number of pairs to be compared – every joint sphere with every environment sphere. $||\vec{t}_j - \vec{t}_s||$ is the distance between those two spheres in the world frame. This methodology for creating ceq_k constraints can be replaced by some more advanced collision check methods [39]. Constraints defined by Equations (33) and (34) are taken for every given pose. They can also be extended to the collision check between particular joints if there is such a risk. The set of optimized variables consists only of the parameters s for 2nd to n^{th} joint. The principle of finding the collision-free position of a physical joint is shown in Figure 8. The proposed solution does change the kinematic parameters for DH or RPYXYZ representations, but it does not change the kinematic structure (placement of the joint axes) of the manipulator, i.e. the kinematic parameters in case of PoE stay the same.

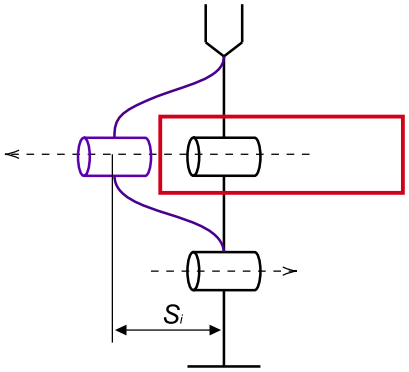


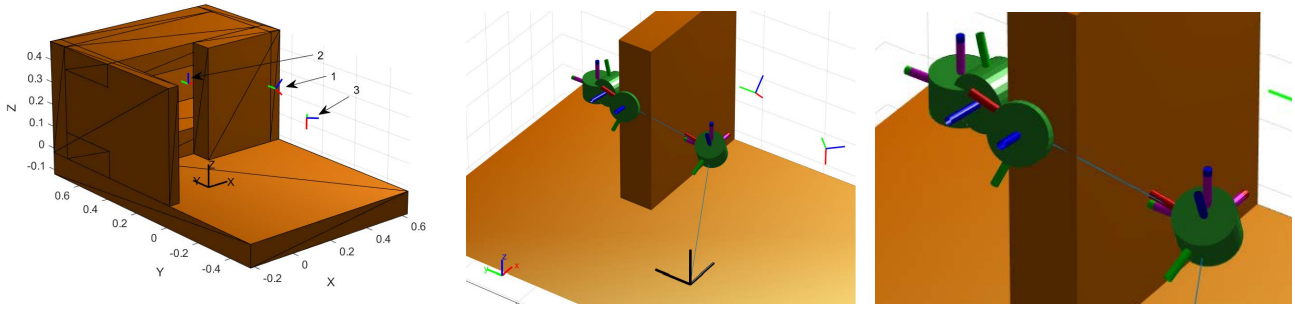
FIGURE 8. Translation of a physical joint along the axis to avoid the collision environment (red).

B. MANIPULATOR DESIGN

Path 1 was chosen with a given collision environment visualized in Figure 9(a). It has three poses:

$$\mathbf{P}_1 = \begin{bmatrix} 0.707 & 0 & 0.707 & 0.45 \\ 0 & 1 & 0 & 0.1 \\ -0.707 & 0 & 0.707 & 0.3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{P}_2 = \begin{bmatrix} 1 & 0 & 0 & 0.3 \\ 0 & 1 & 0 & 0.5 \\ 0 & 0 & 1 & 0.25 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

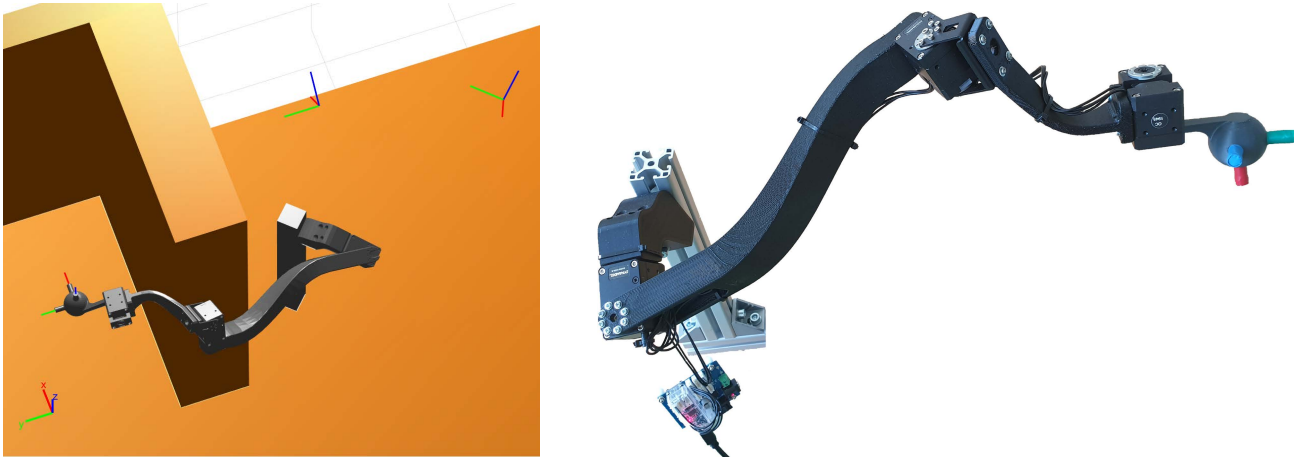


(a) Path 1, the poses are ordered as the number pointing to them show. World frame is black.

(b) Chosen kinematic structure after optimization in the pose 2. The motors and end-effector are represented by cylinders. Some environment walls were hidden for better visibility.

(c) Detail of Figure (b); joint rotation axes Z have blue color, robot base and end-effector frames are purple.

FIGURE 9. Collision environment of the path 1 and optimized kinematic structure.



(a) Simulation of robot design in pose 2; some environment walls were hidden for better visibility.

(b) Experimental setup

FIGURE 10. Collision environment of the path 1 and optimized kinematic structure.

$$\mathbf{P}_3 = \begin{bmatrix} 0 & 0.707 & 0.707 & 0.4 \\ 0 & 0.707 & -0.707 & -0.15 \\ -1 & 0 & 0 & 0.25 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{35}$$

The synthesis was performed, an interval where the base of the synthesized robot can be placed was given in XYZ coordinates as $[0.05, 0.3]$ for X axis, $[0.05, 0.2]$ for Y axis, and $[0.05, 0.3]$ for Z axis. The units are in meters. PoE output of the optimal manipulator given by the algorithm is following:

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0.242 \\ 0 & 1 & 0 & 0.10 \\ 0 & 0 & 1 & 0.24 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{36}$$

$$\mathbf{M} = \begin{bmatrix} 0.235 & 0.282 & 0.930 & -0.231 \\ 0.682 & 0.635 & -0.365 & -0.115 \\ -0.693 & 0.720 & -0.043 & -0.123 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{37}$$

$$\mathcal{S}_1 = \begin{bmatrix} -0.469 \\ -0.149 \\ 0.87 \\ 0 \\ 0 \\ 0 \end{bmatrix}; \ \mathcal{S}_2 = \begin{bmatrix} 0.185 \\ -0.429 \\ 0.884 \\ -0.215 \\ 0.104 \\ 0.095 \end{bmatrix}; \ \mathcal{S}_3 = \begin{bmatrix} 0.157 \\ 0.634 \\ -0.757 \\ 0.217 \\ -0.2 \\ -0.123 \end{bmatrix} \tag{38}$$

where $\mathbf{B}$ is the position of the base frame of the robot, $\mathbf{M}$ is the displacement between the base and the end-effector frame, and $\mathcal{S}_{1..3}$ are rotational screws.

When reaching pose 2, the structure collides with one of the walls of the environment as shown in Figure 9(b).

The methodology described in the previous subsection was applied to find the collision-free position of the joints. The joint position optimization result in a simplified collision environment (spheres only) is visualized in Figure 9(c).

The final robot design is visualized in Figure 10. The links between the joints were manually modeled and 3D printed. The motors Dynamixel XH430-V350-R were chosen for assembly in an experiment to perform the given trajectory. A video of the robot performing the given task is attached to this paper and may be found in the supplementary material. For fast evaluation, the trajectory was planned using the rapidly-exploring random tree (RRT) algorithm implemented in MATLAB, which provided "shaky" behavior at some parts of the path in both simulation and experiment. In future applications, this problem should be addressed.

VII. CONCLUSION

This study expands our understanding of the synthesis of arbitrary kinematic structures of serial robotic manipulators using numerical optimization. It focuses on the performance of two main related problems that were not investigated before: what is the impact of the chosen kinematic representation and what is the impact of the chosen constraint creation method. Thousands of optimization runs were performed for various paths, and a broad analysis is provided and discussed. Any randomness was avoided in the methodology design. The results were supported by the application of two different optimization algorithms, interior point and sequential quadratic programming, with similar outcomes. In the case of robot kinematic representation (choice between Denavit-Hartenberg parameters, Product of Exponentials, and RPYXYZ parameters), the data show that there is no big difference in terms of convergence success rate, but the results differ in optimization time and output manipulator length.

On the other hand, the constraint design method seems to have a major impact on the convergence success rate. In this study, six methods and their combinations were investigated and the success rate differs from 0 to 93 % for the same input. These results shall be verified for other optimization algorithms in future work, including global minimum search methods. Since the implementation of other constraints is not a challenging task, we encourage other researchers who deal with the numerical synthesis of kinematic structures to test their algorithms applying the proposed approach.

In this particular synthesis algorithm, the focus of upcoming research is to expand the objective function to take into account the parameters of velocity or torque between particular poses, as implemented in [24], [25]. The issue is that since the presented approach is a discrete optimization, it is possible that between two consecutive poses the manipulator may need to change configuration.

In addition to the overview on the performance of representations and constraints, this study presents multi-representation and multi-constraint methodologies that can combine the results between each other. The applied optimization algorithm demands an initial estimation of the optimized values, that is, the guess of the robot kinematic structure. Using this multi-representation and multi-constraint approach, it is possible to apply a relatively good result from one optimization, where, for example, DH parameters were searched for, convert them to PoE, and start over with a kinematic structure that is already close to the optimal result, but calculating with a very different approach – matrix exponentials and screw coordinates instead of partial transformations. This increases the convergence success rate and decreases the length of the output manipulator along with the time needed for optimization.

To verify the results presented, an arbitrary manipulator was built, and an experiment was carried out along with the introduction of the split optimization process. The complex task of finding the optimal manipulator design was divided into five steps, where the task of finding the kinematic structure, collision-free performance on a given trajectory, and dynamics analysis was either demonstrated in this paper or referenced in existing studies.

ACKNOWLEDGMENT

The Cooperation Between the authors was initiated thanks to the Aktion Osterreich-Tschechien (AOCZ) Semesterstipendien Grant financed by the Federal Ministry of Education, Science and Research (BMBWF) and organized by the Organized by Austrian Agency for International Cooperation in Education and Research (OeAD-GmbH).

REFERENCES

[1] M. Brandstötter, P. Gallina, S. Seriani, and M. Hofbaur, "Task-dependent structural modifications on reconfigurable general serial manipulators," in *Advances in Service and Industrial Robotics* (Mechanisms and Machine Science). Patras, Greece: Springer, 2018, pp. 316–324.
[2] A. B. Clark and N. Rojas, "Design and workspace characterisation of malleable robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 9021–9027.
[3] Z. Pang, T. Wang, Z. Wang, J. Yu, Z. Sun, and S. Liu, "Design and analysis of a wearable upper limb rehabilitation robot with characteristics of tension mechanism," *Appl. Sci.*, vol. 10, no. 6, p. 2101, Mar. 2020.
[4] A. Zeiaee, R. Soltani-Zarrin, R. Langari, and R. Tafreshi, "Kinematic design optimization of an eight degree-of-freedom upper-limb exoskeleton," *Robotica*, vol. 37, no. 12, pp. 2073–2086, Dec. 2019.
[5] B. M. Otten, R. Weidner, and A. Argubi-Wollesen, "Evaluation of a novel active exoskeleton for tasks at or above head level," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2408–2415, Jul. 2018.
[6] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *J. Appl. Mech.*, vol. 22, no. 2, pp. 215–221, Jun. 1955.
[7] J. J. Uicker, J. Denavit, and R. S. Hartenberg, "An iterative method for the displacement analysis of spatial mechanisms," *J. Appl. Mech.*, vol. 31, no. 2, pp. 309–314, Jun. 1964.
[8] R. S. Ball, "The theory of screws: A study in the dynamics of a rigid body," *Mathematische Annalen*, vol. 9, no. 4, pp. 541–553, Dec. 1876.
[9] K. M. Lynch and F. C. Park, *Modern Robotics*. Cambridge, U.K.: Cambridge Univ. Press, 2017.
[10] M. Joswig and T. Theobald, "Plucker coordinates and lines in space," in *Polyhedral and Algebraic Methods in Computational Geometry* (Universitext). London, U.K.: Springer, 2013, pp. 193–207.
[11] W. Cho, J. Suh, and S.-H. You, "Integrated motion control using a semi-active damper system to improve yaw-roll-pitch motion of a vehicle," *IEEE Access*, vol. 9, pp. 52464–52473, 2021, doi: 10.1109/access.2021.3070366.
[12] M. L. Husty, M. Pfurner, H.-P. Schröcker, and K. Brunnthaler, "Algebraic methods in mechanism analysis and synthesis," *Robotica*, vol. 25, no. 6, pp. 661–675, Nov. 2007.

[13] K. Brunnthaler, H.-P. Schröcker, and M. Husty, "A new method for the synthesis of Bennett mechanisms," in *Proc. Int. Workshop Comput. Kinematics (CK)*, 2005, pp. 1–8.
[14] J. D. Hauenstein, C. W. Wampler, and M. Pfurner, "Synthesis of three-revolute spatial chains for body guidance," *Mechanism Mach. Theory*, vol. 110, pp. 61–72, Apr. 2017.
[15] A. Perez-Gracia and J. M. McCarthy, "Kinematic synthesis of spatial serial chains using Clifford algebra exponentials," *Proc. Inst. Mech. Eng. C, J. Mech. Eng. Sci.*, vol. 220, no. 7, pp. 953–968, Jul. 2006.
[16] R. Pastor, Z. Bobovský, D. Huczala, and S. Grushko, "Genetic optimization of a manipulator: Comparison between straight, rounded, and curved mechanism links," *Appl. Sci.*, vol. 11, no. 6, p. 2471, Mar. 2021.
[17] C. Valsamos, V. C. Moulianitis, and N. Aspragathos, "Metamorphic structure representation: Designing and evaluating anatomies of metamorphic manipulators," in *Advances in Reconfigurable Mechanisms and Robots I*. London, U.K.: Springer, 2012, pp. 3–11.
[18] E. F. Katrantzis, N. A. Aspragathos, C. D. Valsamos, and V. C. Moulianitis, "Anatomy optimization and experimental verification of a metamorphic manipulator," in *Proc. Int. Conf. Reconfigurable Mech. Robots (ReMAR)*, Jun. 2018, pp. 1–7.
[19] S. Ha, S. Coros, A. Alspach, J. M. Bern, J. Kim, and K. Yamane, "Computational design of robotic devices from high-level motion specifications," *IEEE Trans. Robot.*, vol. 34, no. 5, pp. 1240–1251, Oct. 2018.
[20] S. Singh, A. Singla, and E. Singla, "Modular manipulators for cluttered environments: A task-based configuration design approach," *J. Mech. Robot.*, vol. 10, no. 5, Oct. 2018, Art. no. 051010.
[21] S. Patel and T. Sobh, "Task based synthesis of serial manipulators," *J. Adv. Res.*, vol. 6, no. 3, pp. 479–492, May 2015.
[22] E. Singla, S. Tripathi, V. Rakesh, and B. Dasgupta, "Dimensional synthesis of kinematically redundant serial manipulators for cluttered environments," *Robot. Auton. Syst.*, vol. 58, no. 5, pp. 585–595, May 2010.
[23] A. Dogra, S. Sekhar Padhee, and E. Singla, "An optimal architectural design for unconventional modular reconfigurable manipulation system," *J. Mech. Des.*, vol. 143, no. 6, pp. 1–29, Jun. 2021.
[24] J. Whitman and H. Choset, "Task-specific manipulator design and trajectory synthesis," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 301–308, Apr. 2019.
[25] S. Shirafuji and J. Ota, "Kinematic synthesis of a serial robotic manipulator by using generalized differential inverse kinematics," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 1047–1054, Aug. 2019, doi: 10.1109/tro.2019.2907810.
[26] J. Whitman, R. Bhirangi, M. Travers, and H. Choset, "Modular robot design synthesis with deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 6, pp. 10418–10425, doi: 10.1609/aaai.v34i06.6611.
[27] The MathWorks. (2021). *Fmincon*. [Online]. Available: https://Mathworks.com/help/optim/ug/fmincon.html
[28] R. J. Vanderbei and D. F. Shanno, "An interior-point algorithm for nonconvex nonlinear programming," *Comput. Optim. Appl.*, vol. 13, nos. 1–3, pp. 231–252, 1999.
[29] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numer.*, vol. 4, pp. 1–51, Jan. 1995.
[30] P. I. Corke, *Robotics, Vision and Control Fundamental Algorithms in MATLAB*, 2nd ed. Cham, Switzerland: Springer, 2017.
[31] J. Selig, *Geometric Fundamentals of Robotics*. New York, NY, USA: Springer, 2005.
[32] D. Huczala, T. Kot, M. Pfurner, D. Heczko, P. Oščádal, and V. Mostýn, "Initial estimation of kinematic structure of a robotic manipulator as an input for its synthesis," *Appl. Sci.*, vol. 11, no. 8, p. 3548, Apr. 2021.
[33] M. Hofer, "Variational motion design in the presence of obstacles," Ph.D. dissertation, TU Wien, Vienna, Austria, 2004, doi: 20.500.12708/9613.
[34] D. Huczala. (2022). *Robot Manipulator Synthesis—Data and Algorithm*. [Online]. Available: https://zenodo.org/record/6424180
[35] D. Huczala, T. Kot, and M. Pfurner, "An automated conversion between selected robot kinematic representations," 2022, *arXiv:2204.02629*.
[36] T. Kot, Z. Bobovský, M. Brandstötter, V. Krys, I. Virgala, and P. Novák, "Finding optimal manipulator arm shapes to avoid collisions in a static environment," *Appl. Sci.*, vol. 11, no. 1, p. 64, Dec. 2020.
[37] M. Milan, Z. Zdenek, and D. Fojtík, "Automation of the design of the cross-section of the manipulator arms profile," *MM Sci. J.*, vol. 2021, no. 4, pp. 4863–4871, 2021.
[38] Z. Zeman, M. Mihola, and J. Suder, "Design of algorithms for automatic selection of drive units for mechatronic devices," *MM Sci. J.*, vol. 2021, no. 2, pp. 4362–4370, 2021.
[39] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE J. Robot. Autom.*, vol. RA-4, no. 2, pp. 193–203, Apr. 1988.