Robotica: (2021) volume 39, pp. 1760–1778. © The Author(s), 2021. Published by Cambridge University Press.
doi:10.1017/S0263574720001484

Forward Kinematic Modeling of Conical-Shaped Continuum Manipulators

A. H. Bouyom Boutchouang†*, Achille Melingui†, J. J. B. Mvogo Ahanda‡, Othman Lakhal¶, Frederic Biya Motto§ and Rochdi Merzouki¶

†*Department of Electrical and Telecommunications Engineering, Ecole Nationale Supérieure Polytechnique, University of Yaounde I, Yaounde 8390, Cameroon.*
E-mail: achillemelingui@gmail.com

‡*Department of Electrical and Power Engineering, University of Bamenda, Bamenda 39 Bambili, Cameroun.*
E-mail: josephjeanmvogo@yahoo.fr
¶*CRIStAL Laboratory, CNRS-UMR,Villeneuve d'Ascq 59655, France.*
E-mails: lakhal.othman@polytech-lille.fr, rochdi.merzouki@polytech-lille.fr
§*Department of Physic's, Faculty of Sciences, University of Yaounde I, Yaounde 8390, Cameroon.*
E-mail: biyamotto@yahoo.fr

(Accepted December 17, 2020. First published online: February 3, 2021)

SUMMARY
Forward kinematics is essential in robot control. Its resolution remains a challenge for continuum manipulators because of their inherent flexibility. Learning-based approaches allow obtaining accurate models. However, they suffer from the explosion of the learning database that wears down the manipulator during data collection. This paper proposes an approach that combines the model and learning-based approaches. The learning database is derived from analytical equations to prevent the robot from operating for long periods. The database obtained is handled using Deep Neural Networks (DNNs). The Compact Bionic Handling robot serves as an experimental platform. The comparison with existing approaches gives satisfaction.

KEYWORDS: Continuum manipulators; Forward kinematics; Deep learning; Autoencoders (AE) and MLP.

1. Introduction

Continuum manipulators have made the subject of several researches thanks to their capabilities to replicate the behavior of living beings such as muscles,[1] tentacles,[2] elephant trunks,[3,4] and cephalopod members.[5] While conventional manipulators are designed with rigid links, continuum manipulators are made of soft materials that make them flexible and lightweight. This manufacturing makes the continuum manipulators extremely dexterous and offers them the ability to operate in very constraining and delicate environments, such as surgical operations,[6,7] robotic assistance,[8] and exploration.[9,10]

Manipulator kinematics is modeling that highlights the position and orientation of the effector according to variations of configuration space parameters, without taking into account the inertial and vibration effects of the robot. Their deformable structure and the use of different actuators increase

* Corresponding author. E-mail: bouyom_hyacinthe@yahoo.fr

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

the complexity of modeling compared to discrete joint manipulators. Research on the kinematic modeling of continuum manipulators can be classified into two main approaches: model-based and learning-based approaches. Model-based approaches involve establishing the kinematic model, based on the physical structure of the manipulator robot. As for learning-based approaches, the parametric space is divided into several groups according to the robot's operating modes. A mathematical model derived from the learning algorithms makes it possible to establish a relationship between effects (expert observation, sensor measurements, and statistical data) and causes (input references).

A large number of efforts focused on model-based approaches. Hannan and Walker[4] defined the configuration space of a three-dimensional continuum module by three variables using the structural assumption of constant curvature of the backbone.[11] This approximation yields good results thanks to the uniform and symmetrical shape of the manipulator. Godage et al.[12] presented a new three-dimensional kinematic model for multi-section continuum arms using a novel shape function-based approach,[13] which incorporates the constraint of the geometric structure of the arm. Dupont et al.[14] and Escande et al.[15] used the same approach for the forward kinematic modeling of a concentric tube robot and a Compact Bionic Handling Assistant (CBHA), respectively. Similarly, Rolf et al.[16] proposed the forward kinematic model of the Bionic Handling Assistant (BHA) manipulator. Li and Du[17] designed and analyzed a flexible multi-section robot inspired by the snake and octopus's skeletons. The robot consists of three sections, and each section is composed of several identical vertebrae, articulated by spherical joints and a flexible spine. A kinematic model, based on the assumption of a constant curvature piece by piece, was used. However, the contributions mentioned above are based on constant curvature, which may yield imprecise models. For example, the curvature of a conical-shaped continuum robot cannot be constant when it bends. Gravagne et al.[18] and Trivedi et al.[19] proposed the most complete models using the beam and Cosserat rod theories, respectively. These models improve accuracy but are not significant when their calculation and sensing costs are taken into account. Other researchers assumed that the manipulator is a continuous assembly of rigid robots to improve the solution based on the constant curvature. This approach is known as variable curvature (VC). Thus, Chirikjian et al.[20] assumed that the robot consists of a series of a huge number of modules. A shape of a particular mathematical curve is used to solve the kinematics of the manipulator. However, the application is limited, as the method is restricted to a set of reduced shapes and movements to which very few continuous robots can correspond. Jones and Walker[21] developed a kinematic model of a continuum robot with several sections connected in series. Each section is based on a rigid-link arm composed of four pivot joints, two at each end of the section, itself connected by a prismatic joint. Mahl et al.[22] proposed a variable curvature to model BHA manipulator considered as constant curvature (CC) segments. The results are more efficient and can be used for real-time monitoring. Kang et al.[23] modeled a robot segment by four prismatic joints that connect the ends of two consecutive planes and a prismatic joint representing the central length between these two planes. A series of multiple segments are linked to mimic the continuum manipulator's morphology to solve the forward kinematics. Their approach has been applied to a continuum manipulator using pneumatic muscle actuators (PMAs).

Learning-based approaches have been developed to capture the full complexity of continuum deformations and un-modeled nonlinear behaviors of continuum robots. They generally use the sample input–output data pairs to estimate the forward kinematic model (FKM). Melingui et al.[24] proposed a Neural Network with six inputs corresponding to six-wire potentiometer values and three outputs corresponding to parameters of the position of the end-effector to model the forward kinematics of the CBHA. Similarly, Giorelli et al.[25] used a feedforward neural network (FNN) able to represent the relation between the tip position and the forces applied to the cables which drive a non-constant curvature manipulator. Reinhard et al.[26] proposed an artificial neural network with rectified linear unit (ReLU) to solve the forward kinematics of a 6-DOF Concentric Tube Continuum Robot in SE(3) to avoid high computational load and highly nonlinear modeling effort, linked to model-based approaches. To address system uncertainties and improve system robustness, He et al.[27] proposed adaptive neural networks that approximate the unknown model of the robot and adapt robot–patient interactions. Most often, learning-based methods yield accurate models but are very sensitive to sensor noise. Furthermore, the learning database's size increases exponentially with the dimension of the configuration space, and therefore becomes difficult to build for high DoFs manipulators.

An alternative to model the kinematic behaviors of continuum robots is to combine the model-based approach with the learning-based approach to benefit from the best of them. Reinhart et al.[28]

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

combined an analytical model with a learning model taking into account the unmodelled dynamics to feedforward robot control. In the same vein, Lakhal et al.[29] proposed the forward kinematic model of the CBHA robot. They assumed the manipulator as a series of inter-vertebrae, each modeled as a parallel robot. As the inverse kinematic equation (IKE) of an inter-vertebra was highly nonlinear, a multilayer perceptron (MLP) was used to estimate the Cartesian parameters from the configuration space parameters of each inter-vertebra. The successive product of the transformation matrices of each inter-vertebra leads to the position of the end-effector. However, this approach applied to a two-section continuum manipulator may be unsuitable for high DoFs continuum manipulators because of the complexity of the system and the database's size increase. Besides, complex systems with highly nonlinear behaviors and varying can be better estimated with multi-neuron and multilayer networks.[30,31] Thus, shallow architectures generally do not guarantee a good local minimum since gradient descent can easily get stuck in the plateaus of the non-convex learning criterion with depth and a large number of neurons.[32] Furthermore, the number and quality of these local minima and plateaus also influence the chances for random initialization in the basin of attraction (via gradient descent) of a poor solution.[33] Therefore, it is essential to implement learning algorithms of shallow architectures several times to ensure a good convergence. On the contrary, Deep Learning (DL) approach improves the convergence for a highly nonlinear system or having a database representing its complexity since the pre-training process initializes the weights in an area of a good local minimum.[34] For such systems, DL has better convergence than the shallow architectures but does not guarantee the global minima. Deep learning is an approach of training that extracts abstract and complex features of databases using the algorithms with a hierarchical multilevel learning approach. More abstract and complex representations are learned in the higher levels based on the less abstract concepts and representations from the lower level(s) of the learning hierarchy.[35]

Classification[36–39] and regression[40,41] are the main domains of Deep networks whose effectiveness is well established. Thus, a DL-based approach is proposed in this paper to solve the forward kinematics of continuum manipulators. We deal with the case where each section is controlled by many actuators of the same nature and characteristics. The modeling approach is issued from a model-based of parallel rigid robots. Each section is considered by a series of inter-modules modeled as a parallel robot with k universal-prismatic spheric and one universal-prismatic (kUPS-1UP) joints. The kinematic model is deduced from the resolution of the IKEs of each inter-module. IKEs obtained do not allow an analytical solution. Therefore, Deep Neural Networks (DNNs) are used to estimate the solutions of the IKEs of the entire inter-modules. DNNs are generative models and allow the network to converge toward a good local minimum.

The remainder of this paper is organized as follows. Section 2 provides a brief introduction to conical continuum manipulators. The design methodology for FKM of hyper-redundant manipulators is presented in Section 3. The DL-based solution is presented in Section 4. The unsupervised learning of the DNNs based on autoencoder (AE) algorithms, the database generation procedure, and the derivation of the kinematic model of the robot are successively detailed. Section 5 focuses on implementation to the CBHA robot, the validation of the proposed approach, and the comparison with previous approaches. The conclusion and future work are as follows in Section 6.

2. Conical-Shaped Continuum Manipulator

Conical-Shaped Continuum Manipulators (CSCM) are characterized by a curvature that increases along their cross-section and allows them to grasp objects of more varied sizes and shapes, as shown in Fig. 1. This particular characteristic has led to the design of several CSCM inspired by octopus arms or elephant trunk.[16,21,42] Among the methods proposed for modeling the forward kinematics of continuum manipulators, the constant curvature bending approximation is generally used due to its simplification in modeling.[11] Several authors have attempted to use this method for CSCM, but the performance of the models obtained was not satisfactory.[16,43] In this paper, a hybrid approach that combines the model-based and learning-based approaches is proposed for modeling the CSCM.

3. Modeling Approach

This section presents the proposed modeling approach for conical-shaped manipulators. The modeling assumptions are first formulated, and the inverse kinematic equations of an inter-module are then derived.

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

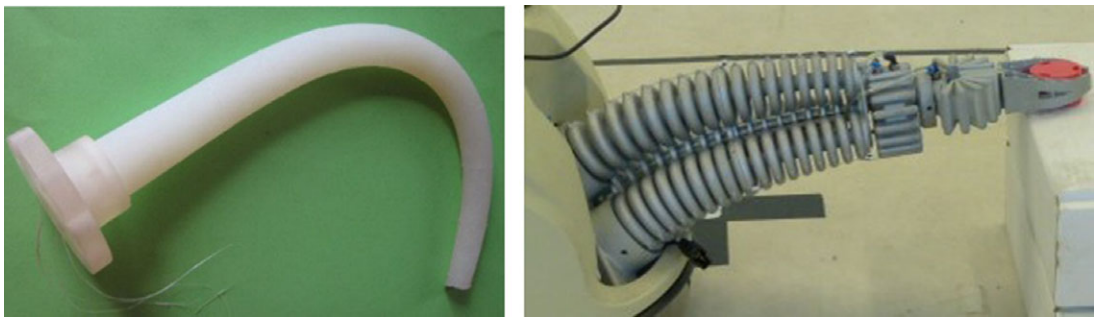


Fig. 1. Examples of conical-shaped continuum manipulators.

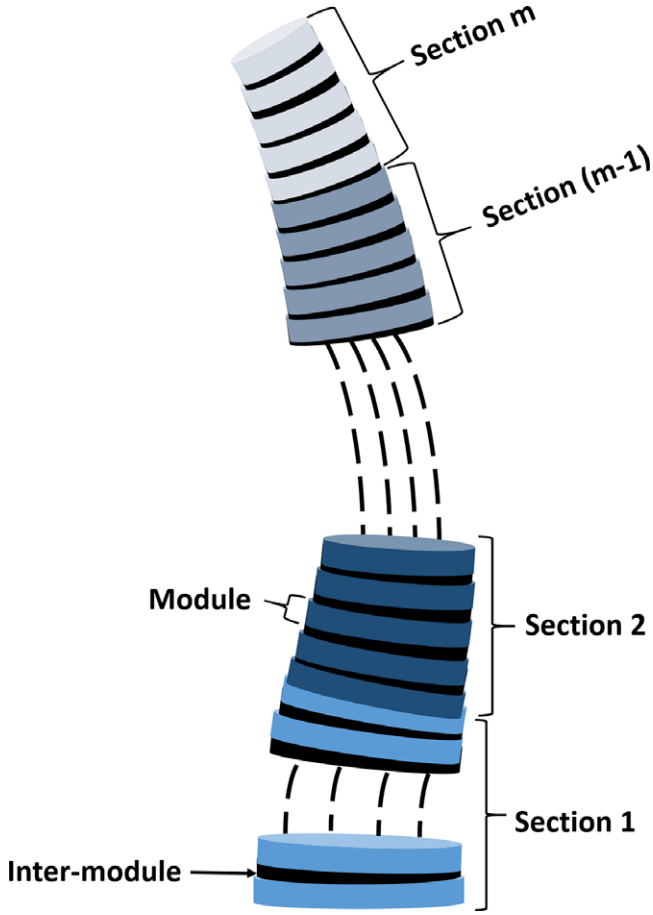


Fig. 2. Structure of a class of conical continuum manipulators. The inter-modules are shown in black while the modules are blue.

3.1. Assumptions

The structure of a CSCM can be represented, as shown in Fig. 2. Some assumptions are considered to model its kinematic behavior:

1. The manipulator is considered as a series of many sections, each controlled independently.
2. A section is considered as a series of many modules.
3. An inter-module is flexible and non-deformable with k-DoFs mobility, as shown in Fig. 3.

Based on these assumptions, an inter-module depicted in Fig. 3 can be considered as a rigid parallel robot composed of k independent kinematic chains whose ends are connected by two rigid

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

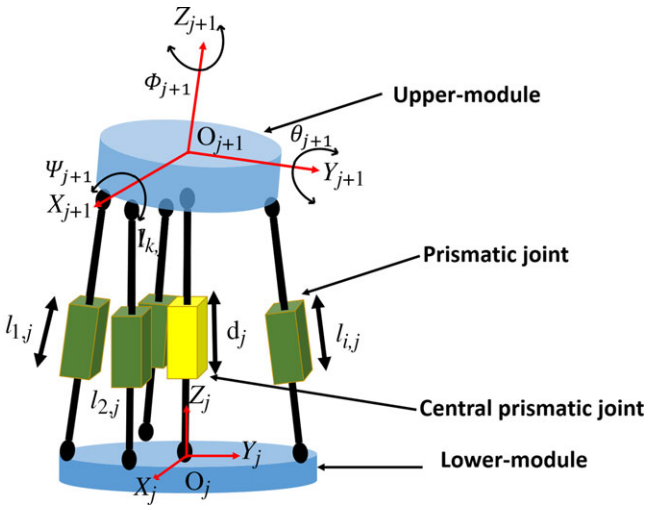


Fig. 3. Inter-module's paramaters in the general case.

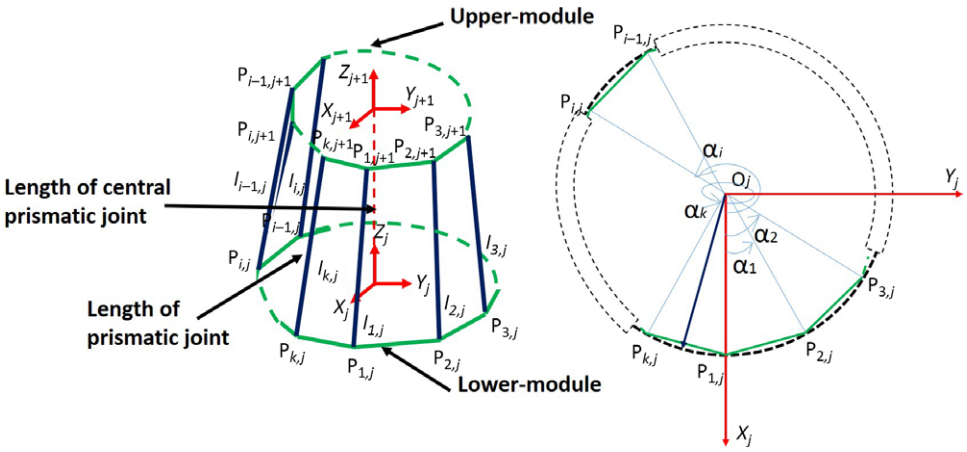


Fig. 4. Spatial configuration of an inter-module.

platforms, namely the lower and upper modules. The lower ends of the joints linked to the lower module form a regular polygon, as shown in Fig. 4. This is also the case for the upper ends linked to the upper module. These joints have variable lengths and are represented by $l_{i,j}$. They provide the position and orientation of the upper module relative to the lower module. The variable represented by d_j is the distance between the centers of gravity of the lower and upper modules; it is considered as a passive joint. The orientation represented by ψ_j, θ_j and ϕ_j are the roll, pitch, and yaw angles, respectively. The inter-module is then modeled as a parallel robot with k universal-prismatic spheric and one universal-prismatic (kUPS-1UP joints).

3.2. Inverse Kinematic Equation of an inter-module

The Inverse Kinematic Equations (IKEs) of an inter-module are obtained by calculating the joint variables $l_{i,j}$ of the parallel robot, where i and j are the number of the modules and the number of the universal-prismatic links, respectively. As far as translation movements are concerned, only translation along the z-axis is possible, because there are no translations along the x and y axes. The translation along the z-axis is denoted by $Trans(z, \mathrm{d}_j)$. The IKEs can, therefore, be formulated as follows:

$$l_{i,j} = f_i(\psi_j, \theta_j, \phi_j, \mathrm{d}_j), \tag{1}$$

where $l_{i,j}$ are the lengths of the universal-prismatic links and ψ_j, θ_j, and ϕ_j are the roll, pitch, and yaw angles, respectively.

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

The endpoints of the module j, the center of the circumscribed circle, and the radius of the circumscribed circle are defined by $P_{i,j}$, O_i, and R_j, respectively. The coordinates of the first endpoint $P_{1,j}$ located on x_j-axis and the origin O_i of the reference frame $\Re_j(O_i, x_j, y_j, z_j)$ are given as follows:

$$P_{1,j}\begin{bmatrix} R_j \\ 0 \\ 0 \end{bmatrix} \text{ and } O_i \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

The coordinates of the endpoint $P_{i,j}$ are obtained by performing a rotation $Rot(z_j, \alpha_i)$. Thus, one has

$$O_i P_{i,j} = Rot(O_j, \alpha_i) O_i P_{1,j}, \; P_{i,j} \begin{bmatrix} R_j \cos\alpha_i \\ R_j \sin\alpha_i \\ 0 \end{bmatrix} \tag{2}$$

with $Rot(O_j, \alpha_i) = \begin{pmatrix} \cos\alpha_i & -\sin\alpha_i & 0 \\ \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and $\alpha_i = \frac{2(i-1)\pi}{k}$.

The coordinates of the endpoint $P_{i,j+1}$ relative to the endpoint $P_{i,j}$ in the reference frame $\Re_j(O_i, x_j, y_j, z_j)$ are given as follows:

$$\begin{bmatrix} L_{i,j} \\ 1 \end{bmatrix} = {}^{j}_{j+1}T \begin{bmatrix} P_{i,j+1} \\ 1 \end{bmatrix} - \begin{bmatrix} P_{i,j} \\ 1 \end{bmatrix}, \tag{3}$$

where ${}^{j}_{j+1}T$ is the transformation matrix of the upper module relative to the lower module, and is expressed as

$${}^{j}_{j+1}T = \begin{pmatrix} r^j_{11} & r^j_{12} & r^j_{13} & 0 \\ r^j_{21} & r^j_{22} & r^j_{23} & 0 \\ r^j_{31} & r^j_{32} & r^j_{33} & d_j \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4}$$

with

$$\begin{aligned}
r^j_{11} &= C\phi_j C\theta_j, \\
r^j_{12} &= -S\phi_j C\psi_j + C\phi_j S\theta_j S\psi_j, \\
r^j_{13} &= S\phi_j S\psi_j + C\phi_j S\theta_j C\psi_j, \\
r^j_{21} &= S\phi_j C\theta_j, \\
r^j_{22} &= C\phi_j C\psi_j + S\phi_j S\theta_j S\psi_j, \\
r^j_{23} &= -C\phi_j S\psi_j + S\phi_j S\theta_j C\psi_j, \\
r^j_{31} &= -S\theta_j, \\
r^j_{32} &= C\theta_j S\psi_j, \\
r^j_{33} &= C\theta_j C\psi_j.
\end{aligned}$$

The symbols S and C denote the sine and cosine functions, respectively. $L_{i,j}$ is a 3×1 matrix. Finally, the prismatic variable $l_{i,j}$ is equal to the distance between the endpoints $P_{i,j+1}$ and $P_{i,j}$.

$$l^2_{i,j} = {L_{i,j}}^T L_{i,j}. \tag{5}$$

Thus, after introducing (2) and (4) into (3), (5) can be rewritten in the format of (1)

$$\begin{aligned}
l_{i,j} = sqrt(&R^2_{j+1} + R^2_j + d^2_j - 2R_{j+1}d_j(\cos\theta_j \sin\psi_j \sin\alpha_i + \sin\theta_j \cos\alpha_i) \\
&- 2R_{j+1}R_j(\cos\theta_j \cos\alpha_i \cos(\alpha_i - \phi_j) + \cos\psi_j \sin\alpha_i \sin(\alpha_i - \phi_j) \\
&+ \sin\theta_j \sin\psi_j \sin\alpha_i \cos(\alpha_i - \phi_j))).
\end{aligned} \tag{6}$$

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

4. FKM of Conical-Shaped Continuum Manipulators

In this section, the FKM of an inter-module is first derived and the FKM of the conical-shaped continuum manipulator is deducted, thereafter. The FKM of an inter-module is obtained by solving the IKEs. For the sake of the real-time implementation, a deep learning architecture has been implemented to numerically solve IKEs.

4.1. FKM of an inter-module: Deep Neural Networks-based solution

The FKM of an inter-module is achieved by solving the IKEs. This comes down to finding the relation between the joint variables $l_{i,j}$ of an inter-module and the parameters of the upper module $(\psi_j, \theta_j, \phi_j, \mathrm{d}_j)$ relative to the lower module frame, and is defined as

$$(\psi_j, \theta_j, \phi_j, \mathrm{d}_j) = f_i^{-1}(l_{i,j}). \tag{7}$$

Equation (6) being highly nonlinear, numerical methods, such as Newton–Raphson, least squares, or symbolic calculation, are generally used to provide an approximate solution. However, for computational time constraints, learning-based methods become an alternative. A deep learning-based solution is implemented in this paper to provide an approximate solution.

Deep Learning(DL) methods aim to learn feature hierarchies with features of higher levels of the hierarchy formed by the composition of the features of lower levels. Automatically learning features at multiple abstraction levels allow a system to learn complex functions, map the input to output directly from data, without relying entirely on human-made features. This is particularly important for higher level abstractions, which humans often do not know how to specify explicitly in terms of raw sensory input.[30] The architecture of DL adopted in this work is a Deep Neural network (DNN) with many levels of nonlinearities allowing it to represent highly nonlinear regression functions such as those obtained in (6). It has the advantage of allowing the initialization of weights in the region close to a good local minimum thanks to pre-training algorithms.[32]

Let $\mathbf{x}$ be the input vector, $h_j^i(\mathbf{x})$ the output of the jth neuron in the ith layer with $i = 0$ the input layer, $i = l + 1$ the output layer, and l the number of hidden layers. The default activation level is determined by the internal bias b_j^i of each unit. The output $h_j^i(\mathbf{x})$ is expressed as follows:

$$h_j^i(\mathbf{x}) = f(a_j^i(\mathbf{x})), \text{ with } a_j^i(\mathbf{x}) = b_j^i + \sum_k W_{jk}^i h_k^{i-1}(\mathbf{x}), \tag{8}$$

where W_{jk}^i denotes the weights matrix between the $(i-1)$th and ith layers, h_k^{i-1} the output of the kth neuron in the $(i-1)$th layer, and $f(.)$ the activation function with $h^0(\mathbf{x}) = \mathbf{x}$.

The output layer is determined in the same way as follows:

$$\mathbf{O}(\mathbf{x}) = \mathbf{f}(\mathbf{a^{l+1}}(\mathbf{x})), \textbf{ with } \mathbf{a^{l+1}}(\mathbf{x}) = \mathbf{b^{l+1}} + \mathbf{W^{l+1}}\mathbf{h^l}(\mathbf{x}). \tag{9}$$

The training of DNNs as a regression model consists of an unsupervised pre-training part and a supervised fine-tuning part, as shown in Fig. 5. In the pre-training part, the training is performed layer by layer in an unsupervised greedy fashion to maximize the likelihood of training samples. Since the backpropagation algorithm is used, the type of hidden units can be a sigmoid (sigm), a hyperbolic tangent (tanh), or an arctangent (atan). Indeed, these activation functions make the backpropagation algorithm stable. Besides, they have an acceptable degree of smoothness and are easily differentiated,[44] unlike a ReLU function, which has a differentiation problem that can lead to a dying ReLU problem.[45] The interested reader is referred to[32] for a detailed description of DNN and autoencoder algorithms.

4.2. Database generation

The deep learning approach has been implemented to numerical solve the IKEs (6) for real-time implementation and prevent the robot from operating for long periods. This subsection describes the database generation procedure. The algorithm shown in Fig. 6 is implemented for this purpose. The principle consists first of all in setting the minimum and maximum limit values of the different parameters θ_j, ψ_j, and d_j. These limits are obtained experimentally. The parameters are then discretized within the defined ranges, and the Eq. (6) is evaluated. Finally, the sets of parameter values θ_j, ψ_j, and d_j whose values of the inter-modules elongations $l_{i,j}$ are within the limits of the latter are saved.

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

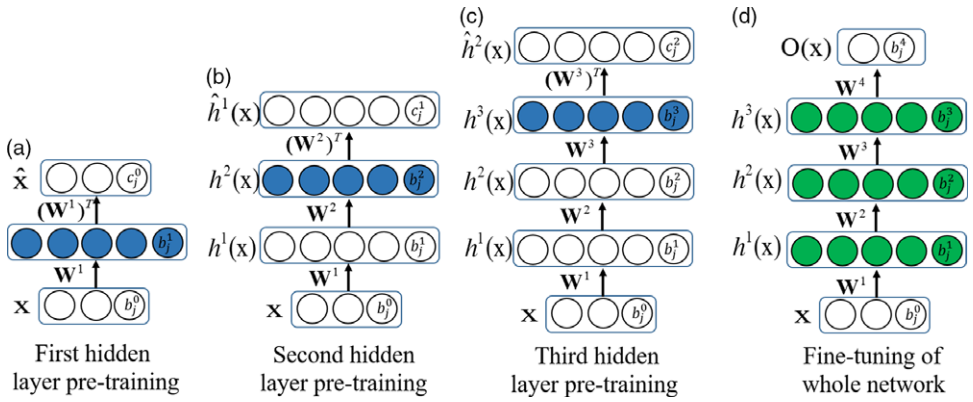


Fig. 5. Training procedure for DNNs using the autoencoder (AE) algorithm for pre-training. (a), (b) and (c) show the pre-training process of the first, second, and third hidden layers, while (d) shows the fine-tuning of the entire network.

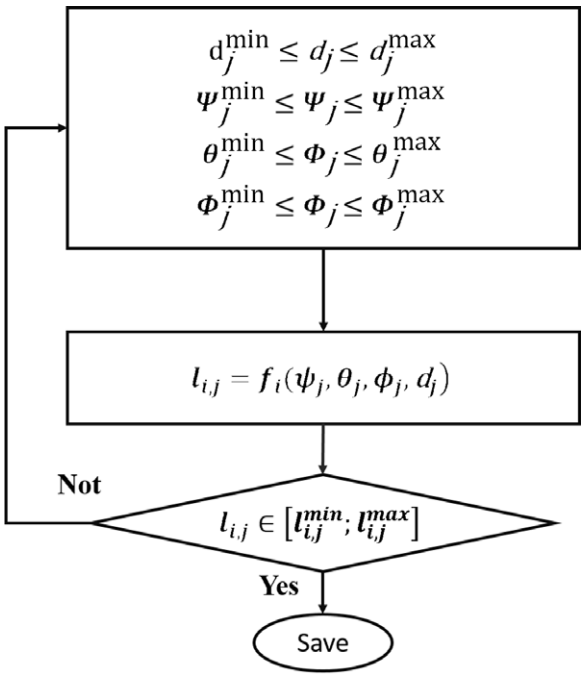


Fig. 6. Generation algorithm of the learning database.

4.3. Optimization of DNN parameters

The hyperparameters of the DNN, namely the number of neurons by hidden layer, the activation function, the learning rate, and the number of epochs are optimized using particle swarm optimization (PSO).[46] Unlike simulated annealing[47] and genetic algorithm,[48] particle swarm optimization ensures an optimal solution in a short period. Besides, simulated annealing requires a much greater number of investigations to produce optimal solutions,[49] while genetic algorithm requires a complex coding and decoding process that increases the number of investigations.[50] The performances achieved by the different algorithms in the case of the pre-training of Deep Neural Network are listed in Table I. We notice that the PSO algorithm achieves the best performance in terms of computation time. Moreover, the side of the neural network architecture is also acceptable.

In the PSO algorithm,[46] the particles are placed in space, and each particle evaluates the objective function according to its current location and the problem to be solved. Then, each particle calculates its displacement in the search space, taking into account its best location as well as those of its neighboring particles, the history of its own current location, and some random perturbations. Once

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

Table I. Results of the optimization algorithms for pre-training process.

Algorithms	Solution finding time in secondes	Performance on validation set	Number of neurons
Genetic	1019	2.0346×10^{-5}	281
Particle swarm	803	2.3879×10^{-5}	256
Simulated annealing	1694	4.8105×10^{-5}	265

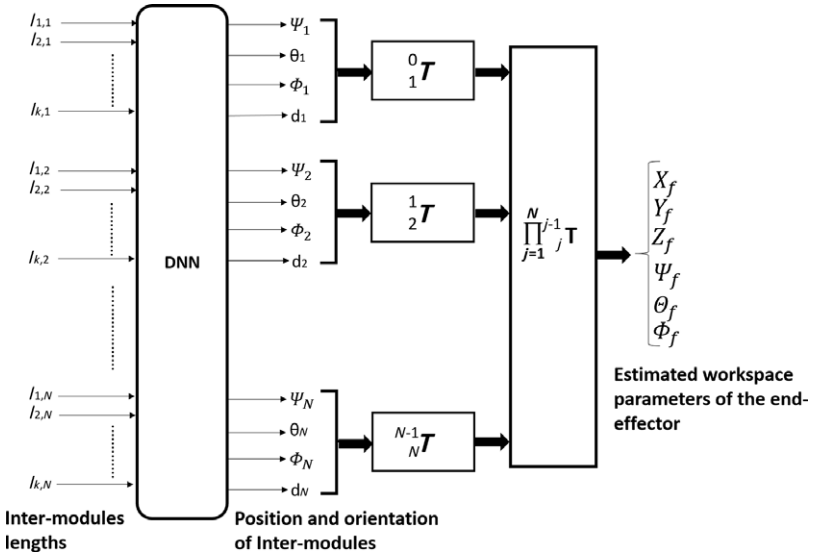


Fig. 7. Computation of FKM in using the DL.

all the particles have been moved, the calculation is repeated. The particles are handled according to the following equations:[51]

$$v_{id} = w \cdot v_{id} + c_1 \cdot rand() \cdot (p_{id} - x_{id}) + c_2 \cdot rand() \cdot (p_{gd} - x_{id}), \tag{10}$$

$$x_{id} = x_{id} + v_{id}, \tag{11}$$

where x_{id} is an element of the ith particle $X^i = \left(x_1^i, x_2^i, \ldots, x_D^i\right)$ with D the dimension of the vector X^i. v_{id} is an element of the rate of the position change (velocity) of the ith particle $V^i = \left(v_1^i, v_2^i, \ldots, v_D^i\right)$. p_{id} is an element of the best previous position of the ith particle $P^i = \left(p_1^i, p_2^i, \ldots, p_D^i\right)$ and p_{gd} the best particle among all the particles in the population. c_1 and c_2 are the acceleration coefficients of v_{id} in the direction of the best p_{id} and p_{gd}. The inertia weight w is employed to control the impact of the previous history of velocities on the current velocity, thereby influencing the trade-off between global (wide-ranging) and local (fine-grained) exploration abilities of the "flying points".

In the case of DNN parameters optimization, the number of neurons per hidden layer, the activation function for each layer, the learning rate, and the number of epochs constitute the elements of the particle position vector X^i. P^i represents the previous best parameter of DNN, and p_{gd} is the best of DNN parameters among all DNN parameters in the population. V^i is formed of the step size of DNN parameters.

4.4. Forward Kinematic Model

The FKM is obtained by following the process depicted in Fig. 7. The joint variables ($l_{i,j}$) of the inter-modules are obtained from the lengths provided by the length sensors of each tube. These joint variables are used as input to the DNNs model, and the end-effector pose is obtained by

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

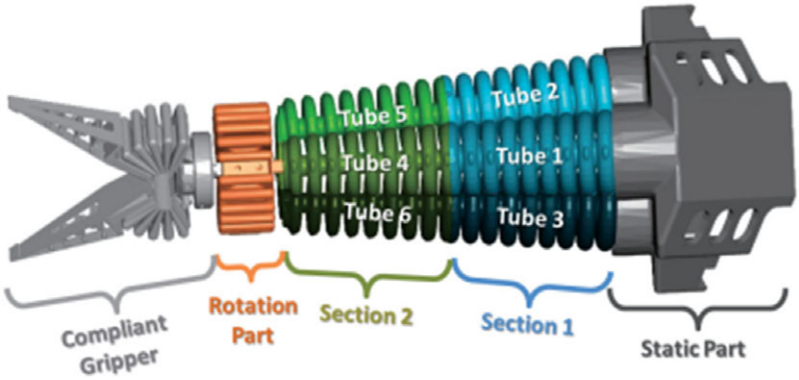


Fig. 8. Compact Bionic Handling Assistant(CBHA).

using the transformation matrices ${}^{j}_{j+1}T$, for $j = 1, ..., N-1$. The composition of homogeneous transformations allows obtaining the end-effector pose as follows:

$$\prod_{j=1}^{N} {}^{j}_{j+1}T = \begin{pmatrix} r_{11}^f & r_{12}^f & r_{13}^f & X_f \\ r_{21}^f & r_{22}^f & r_{23}^f & Y_f \\ r_{31}^f & r_{32}^f & r_{33}^f & Z_f \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{12}$$

with

$$\begin{cases} \Psi_f = \text{atan2}(r_{32}^f, r_{33}^f) \\ \Theta_f = \text{atan2}\left(-r_{31}^f, \sqrt{(r_{11}^f)^2 + (r_{21}^f)^2}\right) \\ \Phi_f = \text{atan2}(r_{21}^f, r_{11}^f) \end{cases} \text{ if } \Theta_f \neq \pm\tfrac{\pi}{2}.$$

and

$$\begin{cases} \Phi_f - sign(\Theta_f)\Psi_f = \text{atan2}(r_{23}^f, r_{13}^f) \\ \quad or \\ \Phi_f - sign(\Theta_f)\Psi_f = -\text{atan2}(r_{23}^f, r_{13}^f) \end{cases} \text{ if } \Theta_f = \pm\tfrac{\pi}{2}$$

5. Application to CBHA Robot

In this section, we focus on the validation of the proposed FKM scheme. It begins with a description of the experimental platform, followed by the database generation process and the results of the training of each architecture. The section concludes with a comparison with the FKM models previously developed, namely model-based and learning-based approaches and the MLP-hybrid approach.

5.1. Experimental platform

The proposed scheme is validated using a Compact Bionic Handling Assistant (CBHA) continuum manipulator, as shown in Fig. 8. It is a compact version of the Bionic Handling Assistant (BHA) robot[16] and is inspired by the elephant's trunk. CBHA is both economical and lightweight, thanks to its additive manufacturing process in which printing is carried out by polyamides. It consists of two bending sections, each equipped with three pneumatic actuators, a wrist axis, and a compliant gripper. A PID regulator controls the pressure supply in each tube. The elongations of the different tubes are provided by six-wire cable potentiometers placed along each tube. Although the CBHA manipulator offers only seven DoFs, its inherent flexibility gives it virtually an infinite number of degrees of freedom.

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

Table II. CBHA's parameters. m, k, and ϕ are the number of sections, the number of prismatic joints of the inter-module, and the yaw angle of the inter-module, respectively. n_1, n_2 denote the number of inter-modules of the first and second sections, respectively.

m	n_1	n_2	k	ϕ
2	8	8	3	0°

Table III. Length limits of inter-modules in (mm).

j	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
$l_{i,j}^{\min}$	8.5	8.5	8.5	7	7	7	8.5	10
$l_{i,j}^{\max}$	19.5	19.5	19.5	16.5	16.5	16.5	20	22
j	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
$l_{i,j}^{\min}$	6.5	8	8	7.5	7.5	7	7	6.5
$l_{i,j}^{\max}$	16	20.5	20.5	19	19	18	18	16

5.2. Database generation

The building of the database is an essential step in the proposed FKM modeling approach. The algorithm presented in Fig. 6 is used for this purpose. According to the algorithm, the parameters of the robot are initialized with large ranges. Then, the IKEs are computed, and the samples of input–output pairs included in the range of the length limits of the inter-modules $l_{i,j}$ are saved. The parameters of the CBHA lists in Table II are obtained following the parameters of the continuum manipulator described in Section 3 and the description of the CBHA robot given above.

Note that the rotation of the inter-module relative to the yaw angle (ϕ_j) is almost null; it is ignored. From (6), the universal-prismatic joint variables of the CBHA robot are expressed as follows:

$$l_{1,j} = sqrt\left(R_{j+1}^2 + R_j^2 + d_j^2 - 2R_{j+1}d_j \sin\theta_j - 2R_{j+1}R_j \cos\theta_j\right), \tag{13}$$

$$l_{2,j} = sqrt\Bigg(R_{j+1}^2 + R_j^2 + d_j^2 - R_{j+1}d_j \left(\sqrt{3}\cos\theta_j \sin\psi_j - \sin\theta_j\right) - R_{j+1}R_j \left(\frac{1}{2}\cos\theta_j + \frac{3}{2}\cos\psi_j - \frac{\sqrt{3}}{2}\sin\theta_j \sin\psi_j\right)\Bigg), \tag{14}$$

$$l_{3,j} = sqrt\Bigg(R_{j+1}^2 + R_j^2 + d_j^2 + R_{j+1}d_j \left(\sqrt{3}\cos\theta_j \sin\psi_j + \sin\theta_j\right) - R_{j+1}R_j \left(\frac{1}{2}\cos\theta_j + \frac{3}{2}\cos\psi_j + \frac{\sqrt{3}}{2}\sin\theta_j \sin\psi_j\right)\Bigg). \tag{15}$$

From (13), (14), and (15), the database of 32,000 samples is generated according to the limit values of the configuration space parameters of each inter-module presented in Table III and the central prismatic joint between 10 and 13 mm.

5.3. FKM of CBHA: Optimization of DNN architecture

As mentioned previously, the PSO algorithm is used to optimize the DNN architecture. The following hyperparameters: the number of neurons by hidden layer, the activation function, the learning rate, and the number of epochs are mainly optimized. Table IV lists the range of variation of the different hyperparameters. The second column represents the limits values of the different parameters, while the third column gives the limit values of the different rates of each hyperparameter. The simulation is performed in Matlab software using the Intel(R) Core(TM) i7-7500U CPU processor at 2.70 GHz and 2.90 GHz. The following activation function, Hyperbolic tangent(tanh),

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

Table IV. Limit values, range of the rates of the hyperparameters using in optimization algorithm.

Parameters	**Limit values of parameters**	**Limits values of the rates**
Neurons	[10; 500]	[−20; 20]
Number of epochs	[100; 1000]	[−100; 100]
Learning for pre-training	[0.005; 0.04]	[−0.01; 0.01]
Learning for training and fine-tuning	[0.2; 0.8]	[−0.1; 0.1]
Activation function	/	[−0.5; 0.5]

Table V. Pre-training results for each hidden layer (four hidden layers) using the Autoencoder algorithm. N_n, N_e, AFHL, and η refer to the number of neurons, the number of epochs per hidden layer, the activation function for the hidden layers, and the learning rate, respectively.

Characteristics					**Mean Square error (MSE)**		
Layer	N_n	N_e	**AFHL**	η	**Training**	**Validation**	**Test**
1st	256	446	sigm	0.028	2.3879×10^{-5}	2.5295×10^{-5}	2.5529×10^{-5}
2nd	234	371	sigm	0.037	2.6820×10^{-6}	2.9724×10^{-6}	3.0033×10^{-6}
3rd	246	404	sigm	0.021	3.5694×10^{-6}	3.8727×10^{-6}	3.9399×10^{-6}
4rd	275	424	sigm	0.015	2.9495×10^{-6}	3.2275×10^{-6}	3.2547×10^{-6}

Table VI. Fine-tuning results for each DNN network: N_{hl}, N_e, AF, and η refer to the number of hidden layers, the number of epochs per hidden layer, the activation function for the output layer, and the learning rate, respectively.

Characteristics				**Mean Square error (MSE)**		
N_{hl}	N_e	**AFOL**	η	**Training**	**Validation**	**Test**
1	471	linear	0.449	8.7650×10^{-5}	8.8667×10^{-5}	8.6362×10^{-5}
2	500	linear	0.673	1.3805×10^{-5}	1.3811×10^{-5}	1.4216×10^{-5}
3	402	linear	0.567	4.3031×10^{-6}	4.6016×10^{-6}	4.7089×10^{-6}
4	431	linear	0.271	7.3149×10^{-6}	7.3777×10^{-6}	7.5069×10^{-6}

sigmoid(sigm), arctangent(atan), and linear function (only for the output layer), are considered in the optimization process. DNN architecture consists of 48 inputs and 48 outputs.

The database is randomly divided into the ratio of 70 : 15 : 15 for training, validation, and test sets. To ensure a proper generalization of the MLP (DNN) models and avoid over-fitting, the early stopping method is applied for the training.[52] The method requires that after some epochs using the training set, the weight matrices of the MLP or DNNs model are fixed, and the latter is operated in the forward mode using the validation set. The process is repeated until the mean square error (MSE) on the validation set reaches its minimum value. The training was stopped when the stopping criterion is fulfilled or when the maximum epochs occurred.

5.3.1. DNN architecture: Pre-training and fine-tuning results. Table V lists the optimized parameters obtained in the pre-training process using the Autoencoder algorithm. We observe that the number of neurons in different layers is less than 300, and the sigmoid function is selected as the best activation function for all the hidden layers. The MSE on the test set is approximately 10^{-5}.

The performance obtained for each deep neural network model after the fine-tuning process is given in Table VI. We notice that while the sigmoid activation function has been selected for hidden layers, the linear activation function is selected by the optimization algorithm for the output layer. We also notice that a DNN achieves the best performance consisting of 3 hidden layers with 256, 234, and 246 in the first, second, and third hidden layers.

5.3.2. Analysis of the performances of DNN and MLP architectures. For the sake of motivating in the choice of DNN architecture, the PSO algorithm has also been applied to shallow architecture that consists of one, two, three, and four hidden layers. The results obtained are given in Table VII. Like

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

Table VII. MLP training result: N_{hl}, N_n, N_e, and η refer to the number of hidden layers, the number of neurons per hidden layers, the number of epochs, and the learning rate, respectively. AFHL and AFOL denote the activation function of hidden and output layers.

Characteristics	N_{hl}	1	2	3	4
	N_n	103	[93; 83]	[89; 107; 93]	[92; 106; 72; 85]
	AFHL	sigm	[sigm; sigm]	[sigm; sigm; sigm]	[sigm; sigm; sigm; sigm]
	AFOL	linear	linear	linear	linear
	N_e	356	429	488	322
	η	0.307	0.313	0.366	0.313
MSE	Training	4.5797×10^{-4}	4.6266×10^{-5}	4.6995×10^{-5}	5.8980×10^{-5}
	Validation	4.5566×10^{-4}	4.7404×10^{-5}	4.6028×10^{-5}	5.9529×10^{-5}
	Test	4.6222×10^{-4}	4.5797×10^{-5}	4.6141×10^{-5}	5.8456×10^{-5}

Table VIII. Characteristics of DNN architectures for one, two, three, and four sections with three hidden layers. m is the number of sections and N_{im} is the number of inter-modules. N_{ept} and N_{eft} are the number of epochs of pre-training and fine-tuning, respectively. η_{ept} and η_{eft} are the learning rates of pre-training and fine-tuning, respectively.

Characteristics	m	1	2	3	4
	N_{im}	8	16	24	32
	N_n	[127; 120; 153]	[256; 234; 246]	[290; 304; 293]	[406; 397; 380]
	N_{ept}	[808; 610; 786]	[446; 371; 404]	[300; 297; 333]	[361; 390; 218]
	AFHL	[sigm; sigm; sigm]	[sigm; sigm; sigm]	[sigm; sigm; sigm]	[sigm; sigm; sigm]
	η_{pt}	[0.034; 0.029; 0.035]	[0.028; 0.037; 0.021]	[0.009; 0.013; 0.029]	[0.022; 0.036; 0.017]
	N_{eft}	464	402	471	475
	AFOL	linear	linear	linear	linear
	η_{ft}	0.643	0.567	0.636	0.337
MSE	Training	1.8850×10^{-6}	4.3031×10^{-6}	8.9648×10^{-6}	3.1650×10^{-5}
	Validation	1.9277×10^{-6}	4.6016×10^{-6}	8.8015×10^{-6}	3.5377×10^{-5}
	Test	2.0592×10^{-6}	4.7089×10^{-6}	8.9759×10^{-6}	3.6480×10^{-5}

DNN architecture, the sigmoid and linear functions are also selected as activation functions for hidden layers and an output layer. However, the best performance is achieved by shallow networks that sconsist of 2 hidden layers with 93 and 83 neurons in the first and second hidden layer, respectively. An analysis of the performance achieved by the DNN and MLP shows that the different architectures can accurately predict the parameters of the inter-modules of the CBHA robot. However, DNN achieved the best MSE on the test set despite the high number of neurons in the different hidden layers. The performance of MLP in the case of CBHA modeling is acceptable, and the size of the architecture is better than that of DNNs. Although the shallow networks achieve the acceptable performance for a two sections continuum robot, this may be deteriorated as the number of sections increases.

5.3.3. Evolution of DNN architecture with the number of sections. This subsection investigates the complexity of the DNN models with an increase in the number of sections (or modules). Therefore, the PSO algorithm is implemented to continuum manipulators successively that consist of 1 (8 inter-modules), 2 (16 inter-modules), 3 (24 inter-modules), and 4 (32 inter-modules) sections. Table VIII gives the results obtained. The sigmoid and linear activation functions have been selected for hidden layers and output layer, respectively. We observed that the complexity of DNNs architecture increases with the number of sections. The number of neurons in different hidden layers for four sections is approximately three times the number of hidden neurons for one section. Besides, the MSE decreases with an increase in the number of sections.

5.4. FKM validation

The FKM is validated by comparing the poses obtained by the proposed FKM with those provided by the Optitrack motion capture system[53] depicted in Fig. 9. A comparison is conducted on the

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

Table IX. Inter-module's lengths as a function of the lengths of tubes.

j	1	2	3	4	5	6	7	8
proportion	0.13	0.13	0.13	0.11	0.11	0.11	0.13	0.15
j	9	10	11	12	13	14	15	16
proportion	0.11	0.14	0.14	0.13	0.13	0.12	0.12	0.11

Table X. Mean values and variance of the euclidean error of the workspace parameters (Position in mm and orientation in °).

Networks	**Axis**	X	Y	Z	θ	ϕ	ψ
MLP (Lakhal et al.[29])	Mean value	−2.142	0.298	−3.158	−1.541	−0.303	−1.729
	Variance	1.859	1.990	1.014	1.423	0.193	2.081
MLP	Mean value	−2.601	1.837	−3.923	1.074	−0.252	−1.327
	Variance	1.781	1.287	2.107	6.724	0.467	7.954
DNN	Mean value	−0.765	0.186	−0.710	−0.231	0.189	0.686
	Variance	0.307	0.449	0.598	0.556	0.050	0.447

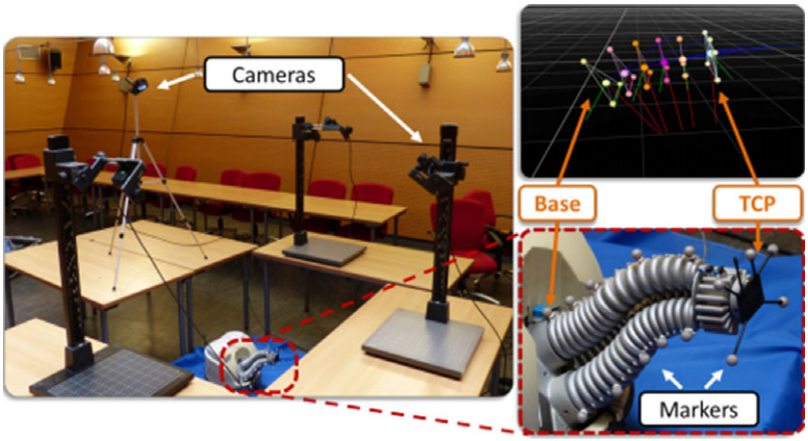


Fig. 9. Experimental scheme of FKM validation.

entire CBHA's workspace. The motion capture system consists of four cameras that can track a rigid body's pose with an accuracy of ±0.1 mm. The CBHA's workspace is explored by supplying the pressure in the six CBHA's tubes. The input vectors are obtained by discretizing the pressure variable with a step size of 0.5 bar. An input vector $[p_1, p_2, \ldots, p_6]$, with $p_i \in [0.0, 0.5, 1.0, 1.5, 2.0]$ and $i = 1, 2, \ldots, 6$ is supplied to CBHA's tubes. This results in a 15, 625 samples database and a robot operating time of 13 h. The samples are recorded after a complete mechanical equilibrium of the robot, which takes approximately 3 s. For each input pressure vector, the end-effector pose, and the corresponding wire potentiometer voltages are recorded. These wire potentiometer voltages are then transformed into lengths using a simple linear transformation function. Finally, the lengths of each inter-module are derived by using the proportionality coefficients presented in Table IX. These proportionality coefficients are obtained empirically, after a series of some experiments on the CBHA. This experiment consists of recording for each position of the end-effector, the lengths of the tubes, and the lengths of the inter-modules. Then, calculate the coefficient of proportionality only between the tubes lengths and the lengths of the inter-modules which are linked. The lengths of the inter-modules are fed to the proposed FKM to estimate the end-effector positions and orientations.

The most frequently used statistical tools for data analysis are mean value and variance. The Euclidean error is the difference between the estimated and desired value. The results of the Euclidean error are presented in the form of a histogram, as shown in Figs. 10 and 11. These plots show that the position and orientation errors do not exceed ±5 mm and ±7° for DNN, ±8 mm, and ±12° for one MLP per section, and ±8.5 mm and ±13° for one MLP for both sections.

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

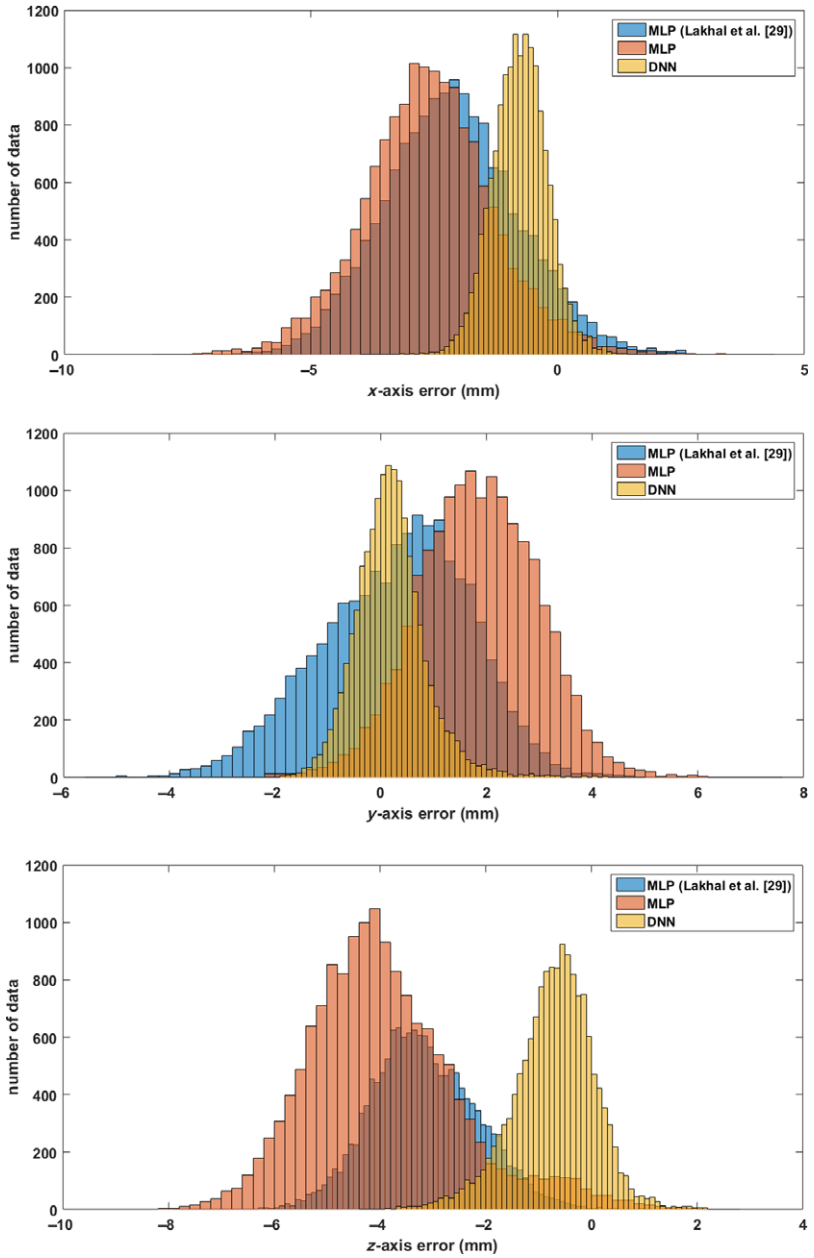


Fig. 10. Comparison between the MLP(Lakhal et al.[29]), MLP and DNN on the position of the end-effector.

Table X shows the mean value and the variance on the errors of each parameter. Satisfactory performance is noted with the mean absolute values of the position and orientation errors less than ± 1 mm and $\pm 1^\circ$ for DNN, ± 3.5 mm and $\pm 2^\circ$ for one MLP per section (Lakhal et al.[29]), and ± 4 mm and $\pm -1.5^\circ$ for one MLP for two sections. Besides, the variance shows good generalization for DNN and MLP (Lakhal et al.[29]) with variance values less than 0.6 and 2, respectively. However, the generalization is poor performance for the MLP architecture (one MLP for both sections), especially in the orientation where the variance is approximately 7.

5.5. Discussions

Based on the results obtained, the MLP network gives the best performance with two hidden layers and loses convergence with an increase in the number of hidden layers. Besides, the MLP network shows imprecision for high-sections continuum manipulators. In contrast, we observe that DNNs give a better performance whatever the number of hidden layers and show their capacity to approximate

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

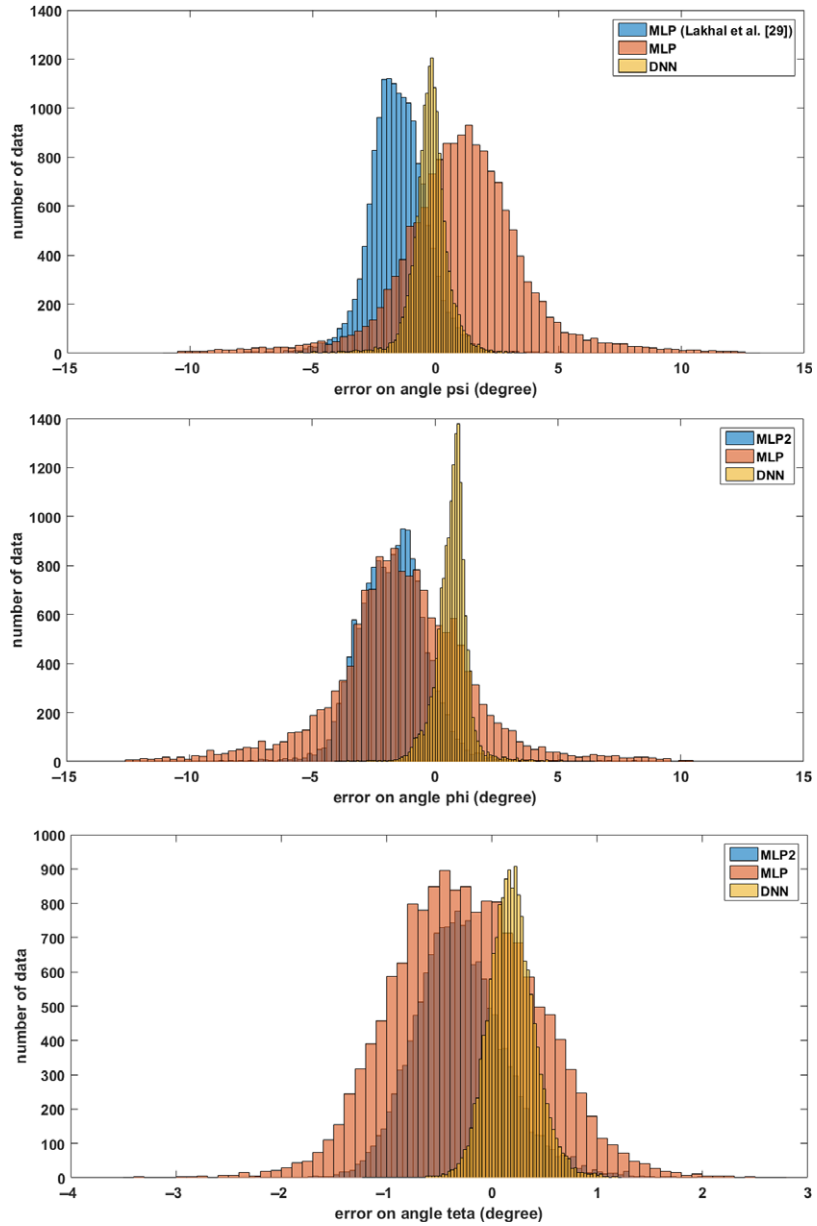


Fig. 11. Comparison between the MLP(Lakhal et al.[29]), MLP and DNN on the orientation of the end-effector.

the workspace parameters of the inter-modules. The same performance is also obtained when the number of inter-modules increases. Furthermore, the DNNs with three hidden layers achieved the best performance. However, the number of neurons varies considerably when the number of sections increases, and the network loses its accuracy progressively.

In summary, based on the results obtained (Table XI), we can note that the model-based approach[43] leads to less accurate models due mainly to many modeling assumptions, such that constant curvature and toroidal deformations. Learning-based approach[24] appears as the most accurate approach because it is free from assumptions. It directly establishes the relationship between Cartesian space and actuation space through a set of nonlinear functions. However, it is generally challenging to derive well-structured control laws based on learning models. Besides, more the degree of freedom of the manipulator increases, more difficult becomes the building of the learning database. The proposed hybrid DL-based approach, which combines the advantages of the model-based and Learning-based approaches, gives acceptable performance; the teaching process is relatively easy as it does not require long periods of robot use.

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

Table XI. Recent contributions on the kinematic modeling of CBHA.

Robots	Characteristics	Modeling method	Kinematic modeling
CBHA	Continuous Shape, Structure: Soft, 2 sections, 3 tubes per section, 17 vertebras, 3 DoFs per inter-vertebra, Pneumatic actuation, Length: 0.50 m	Model-based approach: constant curvature Escande et al.[43]	Forward (FKM), accuracy in position : 8 mm, accuracy in orientation: 23°; Validation: Simulation and real-time experiments
		Learning-based approach: NN, Melingui et al.[24]	Forward (FKM), accuracy: 4 mm; Validation: Simulation and real-time experiments
		Hybrid approach: variable curvature and NN, Lakhal et al.[29]	Forward (FKM), accuracy: 8 mm; Validation: Simulation and real-time experiments
		Hybrid approach: variable curvature and DL, Proposed approach	Forward (FKM), Forward (FKM), accuracy in position : 5 mm, accuracy in orientation: 7°; Validation:real-time experiments

6. Conclusion

This paper proposes a forward kinematic modeling approach of conical-shaped continuum manipulators. For real-time implementation, the proposed approach includes learning-based methods, particularly in solving the resulting inverse kinematic equations. The FKM obtained is computationally inexpensive and can be easily implemented in real-time. It can estimate the pose of the end-effector of continuum manipulators with an acceptable degree of accuracy. The validation of the FKM using an Optitrack motion capture system demonstrates the effectiveness of the proposed modeling approach. In future work, the inverse kinematics and dynamic behavior of the CBHA in interaction with the external environment will be considered for control purposes.

References

1. M. B. Pritts and C. D. Rahn, "Design of an Artificial Muscle Continuum Robot," *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04*, vol. 5 (IEEE, 2004) pp. 4742–4746.
2. I. D. Walker, "Continuous Backbone 'Continuum' Robot Manipulators," *ISRN Robotics*, vol. 2013 (2013).
3. R. Cieślak and A. Morecki, "Elephant trunk type elastic manipulator-a tool for bulk and liquid materials transportation," *Robotica* **17**(1), 11–16 (1999).
4. M. W. Hannan and I. D. Walker, "Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots," *J. Field Robot.* **20**(2), 45–63 (2003).
5. W. McMahan, B. Jones, I. Walker, V. Chitrakaran, A. Seshadri and D. Dawson, "Robotic Manipulators Inspired by Cephalopod Limbs," *Proceedings of CDEN Design Conference* Citeseer, vol. **20**(2) (2004) pp. 1–10
6. D. B. Camarillo, C. R. Carlson and J. K. Salisbury, "Configuration tracking for continuum manipulators with coupled tendon drive," *IEEE Trans. Robot.* **25**(4), 798–808 (2009).
7. N. Simaan, R. Taylor and P. Flint, "A Dexterous System for Laryngeal Surgery," *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 1 (IEEE, 2004) pp. 351–357.
8. J. Casper and R. R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center," *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **33**(3), 367–385 (2003).
9. I. D. Walker, R. Mattfeld, A. Mutlu, A. Bartow and N. Giri, "A Novel Approach to Robotic Climbing Using Continuum Appendages in In-Situ Exploration," *2012 IEEE Aerospace Conference* (IEEE, 2012) pp. 1–9.
10. S. Liu, Z. Yang, Z. Zhu, L. Han, X. Zhu and K. Xu, "Development of a dexterous continuum manipulator for exploration and inspection in confined spaces," *Ind. Robot. Int. J.* **43**(3), 284–295 (2016).
11. R. J. Webster III and B. A. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review," *Int. J. Robot. Res.* **29**(13), 1661–1683 (2010).
12. I. S. Godage, E. Guglielmino, D. T. Branson, G. A. Medrano-Cerda and D. G. Caldwell, "Novel Modal Approach for Kinematics of Multisection Continuum Arms," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2011) pp. 1093–1098.
13. I. S. Godage, D. T. Branson, E. Guglielmino, G. A. Medrano-Cerda and D. G. Caldwell, "Shape Function-Based Kinematics and Dynamics for Variable Length Continuum Robotic Arms," *2011 IEEE International Conference on Robotics and Automation* (IEEE, 2011) pp. 452–457.

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

14. P. E. Dupont, J. Lock and E. Butler, "Torsional Kinematic Model for Concentric Tube Robots," *2019 IEEE International Conference on Robotics and Automation* (IEEE, 2009) pp. 3851–3858.
15. C. Escande, R. Merzouki, P. M. Pathak and V. Coelen, "Geometric Modelling of Multisection Bionic Manipulator: Experimental Validation on Robotinoxt," *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (IEEE, 2012) pp. 2006–2011.
16. M. Rolf and J. J. Steil, "Constant Curvature Continuum Kinematics as Fast Approximate Model for the Bionic Handling Assistant," *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2012) pp. 3440–3446.
17. Z. Li and R. Du, "Design and analysis of a bio-inspired wire-driven multi-section flexible robot," *Int. J. Adv. Robot. Syst.* **10**(4), 209 (2013).
18. I. A. Gravagne, C. D. Rahn and I. D. Walker, "Large deflection dynamics and control for planar continuum robots," *IEEE/ASME Trans. Mech.* **8**(2), 299–307 (2003).
19. D. Trivedi, A. Lotfi and C. D. Rahn, "Geometrically exact models for soft robotic manipulators," *IEEE Trans. Robot.* **24**(4), 773–780 (2008).
20. G. S. Chirikjian and J. W. Burdick, "A modal approach to hyperredundant manipulator kinematics," *IEEE Trans. Robot. Autom.* **10**(3), 343–354 (1994).
21. B. A. Jones and I. D. Walker, "Kinematics for multisection continuum robots," *IEEE Trans. Robot.* **22**(1), 43–55 (2006).
22. T. Mahl, A. Hildebrandt and O. Sawodny, "A variable curvature continuum kinematics for kinematic control of the bionic handling assistant," *IEEE Trans. Robot.* **30**(4), 935–949 (2014).
23. R. Kang, E. Guglielmino, D. T. Branson and D. G. Caldwell, "Kinematic Model and Inverse Control for Continuum Manipulators," *2013 10th IEEE International Conference on Control and Automation (ICCA)* (IEEE, 2013) pp. 1615–1620.
24. A. Melingui, R. Merzouki, J. B. Mbede, C. Escande, B. Daachi and N. Benoudjit, "Qualitative Approach for Inverse Kinematic Modeling of a Compact Bionic Handling Assistant Trunk," *2014 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2014) pp. 754–761.
25. M. Giorelli, F. Renda, G. Ferri and C. Laschi, "A Feed-Forward Neural Network Learning the Inverse Kinetics of a Soft Cable-Driven Manipulator Moving in Three-Dimensional Space," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2013) pp. 5033–5039.
26. R. Grassmann, V. Modes and J. Burgner-Kahrs, "Learning the Forward and Inverse Kinematics of a 6-DOF Concentric Tube Continuum Robot in SE(3)," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2018) pp. 5125–5132.
27. W. He, S. S. Ge, Y. Li, E. Chew and Y. S. Ng, "Neural network control of a rehabilitation robot by state and output feedback," *J. Intell. Robot. Syst.* **80**(1), 15–31 (2015).
28. R. Reinhart, Z. Shareef and J. Steil, "Hybrid analytical and data-driven modeling for feed-forward robot control," *Sensors* **17**(2), 311 (2017).
29. O. Lakhal, A. Melingui and R. Merzouki, "Hybrid approach for modeling and solving of kinematics of a compact bionic handling assistant manipulator," *IEEE/ASME Trans. Mech.* **21**(3), 1326–1335 (2016).
30. Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.* **2**(1), 1–127 (2009).
31. Y. Bengio and Y. LeCun, "Scaling learning algorithms towards AI," *Large-Scale Kernel Mach.* **34**(5), 1–41 (2007).
32. H. Larochelle, Y. Bengio, J. Louradour and P. Lamblin, "Exploring strategies for training deep neural networks," *J. Mach. Learn. Res.* **10**(1), 1–40 (2009).
33. K. Fukumizu and S.-I. Amari, "Local minima and plateaus in hierarchical structures of multilayer perceptrons," *Neural Networks* **13**(3), 317–327 (2000).
34. H. Larochelle, D. Erhan, A. Courville, J. Bergstra and Y. Bengio, "An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation," *Proceedings of the 24th International Conference on Machine Learning* (2007) pp. 473–480.
35. R. Salakhutdinov, "Learning deep generative models," *Ann. Rev. Stat. Appl.* **2**(1), 361–385 (2015).
36. H. Mohsen, E.-S. A. El-Dahshan, E.-S. M. El-Horbaty and A.-B. M. Salem, "Classification using deep learning neural networks for brain tumors," *Future Comput. Inform. J.* **3**(1), 68–71 (2018).
37. A. Kamilaris and F. X. Prenafeta-Bold, "Deep learning in agriculture: A survey," *Comput. Electron. Agricul.* **147**(1), 70–90 (2018).
38. G. Cheng, C. Yang, X. Yao, L. Guo and J. Han, "When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative CNNS," *IEEE Trans. Geosci. Remote Sens.* **56**(5), 2811–2821 (2018).
39. Q. Liu, R. Hang, H. Song and Z. Li, "Learning multiscale deep features for high-resolution satellite image scene classification," *IEEE Trans. Geosci. Remote Sens.* **56**(1), 117–126 (2017).
40. X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan and G. Amaratunga, "Ensemble Deep Learning for Regression and Time Series Forecasting," *2014 IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL)* (IEEE, 2014) pp. 1–6.
41. Q. Wang, J. Wan and Y. Yuan, "Deep metric learning for crowdedness regression," *IEEE Trans. Circ. Syst. Video Tech.* **28**(10), 2633–2643 (2018).
42. F. Renda, M. Cianchetti, M. Giorelli, A. Arienti and C. Laschi, "A 3D steady-state model of a tendon-driven continuum soft manipulator inspired by the octopus arm," *Bioinspiration Biomimetics* **7**(2), 025006 (2012).
43. C. Escande, T. Chettibi, R. Merzouki, V. Coelen and P. M. Pathak, "Kinematic calibration of a multisection bionic manipulator," *IEEE/ASME Trans. Mechatronics* **20**(2), 663–674 (2014).

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press

44. J. Han and C. Moraga, "The Influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning," *International Workshop on Artificial Neural Networks* (Springer, 1995) pp. 195–201.
45. Z. Hu, Y. Li and Z. Yang, "Improving Convolutional Neural Network Using Pseudo Derivative ReLU, "*2018 5th International Conference on Systems and Informatics (ICSAI)* (IEEE, 2018) pp. 283–287.
46. R. Poli, J. Kennedy and T. Blackwell, "Particle swarm optimization," *Swarm Intell.* **1**(1), 33–37, Springer (2007).
47. A. Dekkers and E. Aarts, "Global optimization and simulated annealing" *Math. Program.* **50**(1–3), 367–393, Springer (1991).
48. R. Tanese, *Distributed Genetic Algorithms for Function Optimization* (University of Michigan,1989).
49. S. A. Ethni, B. Zahawi, D. Giaouris and P. P. Acarnley, "Comparison of Particle Swarm and Simulated Annealing Algorithms for Induction Motor Fault Identification," *2009 7th IEEE International Conference on Industrial Informatics* (IEEE, 2009) pp. 470–474.
50. V. Savsani, R. V. Rao and D. P. Vakharia, "Optimal weight design of a gear train using particle swarm optimization and simulated annealing algorithms," *Mech. Mach. Theory* **45**(3), 531–541, Elsevier (2010).
51. R. C. Eberhart and Y. Shi, "Comparison Between Genetic Algorithms and Particle Swarm Optimization," *International Conference on Evolutionary Programming* (Springer, 1998) pp. 611–616.
52. L. Prechelt, Proben1: A set of neural network benchmark problems and benchmarking rules (1994).
53. OptiTrack, Motion capture systems (2018), visited on 06 December 2018. [Online]. Available at: https://optitrack.com/motion-capture-robotics/.

https://doi.org/10.1017/S0263574720001484 Published online by Cambridge University Press