



A Recurrent Neural-Network-Based Real-Time Dynamic Model for Soft Continuum Manipulators

Abbas Tariverdi^{1*}, Venkatasubramanian Kalpathy Venkiteswaran², Michiel Richter², Ole J. Elle^{3,4}, Jim Tørresen⁴, Kim Mathiassen⁵, Sarthak Misra^{2,6} and Ørjan G. Martinsen^{1,7}

¹Department of Physics, University of Oslo, Oslo, Norway, ²Department of Biomechanical Engineering, University of Twente, Enschede, Netherlands, ³The Intervention Centre, Oslo University Hospital, Oslo, Norway, ⁴Department of Informatics, University of Oslo, Oslo, Norway, ⁵Department of Technology Systems, University of Oslo, Oslo, Norway, ⁶Department of Biomedical Engineering, University of Groningen and University Medical Centre Groningen, Groningen, Netherlands, ⁷Department of Clinical and Biomedical Engineering, Oslo University Hospital, Oslo, Norway

OPEN ACCESS

Edited by:

Thomas George Thuruthel,
University of Cambridge,
United Kingdom

Reviewed by:

Marc Daniel Killpack,
Brigham Young University,
United States
Utku Culha,
Max Planck Institute for Intelligent
Systems, Germany

*Correspondence:

Abbas Tariverdi
abbast@uio.no

Specialty section:

This article was submitted to
Soft Robotics,
a section of the journal
Frontiers in Robotics and AI

Received: 19 November 2020

Accepted: 05 February 2021

Published: 18 March 2021

Citation:

Tariverdi A, Venkiteswaran VK,
Richter M, Elle OJ, Tørresen J,
Mathiassen K, Misra S and
Martinsen ØG (2021) A Recurrent
Neural-Network-Based Real-Time
Dynamic Model for Soft
Continuum Manipulators.
Front. Robot. AI 8:631303.
doi: 10.3389/frobt.2021.631303

This paper introduces and validates a real-time dynamic predictive model based on a neural network approach for soft continuum manipulators. The presented model provides a real-time prediction framework using neural-network-based strategies and continuum mechanics principles. A time-space integration scheme is employed to discretize the continuous dynamics and decouple the dynamic equations for translation and rotation for each node of a soft continuum manipulator. Then the resulting architecture is used to develop distributed prediction algorithms using recurrent neural networks. The proposed RNN-based parallel predictive scheme does not rely on computationally intensive algorithms; therefore, it is useful in real-time applications. Furthermore, simulations are shown to illustrate the approach performance on soft continuum elastica, and the approach is also validated through an experiment on a magnetically-actuated soft continuum manipulator. The results demonstrate that the presented model can outperform classical modeling approaches such as the Cosserat rod model while also shows possibilities for being used in practice.

Keywords: continuum manipulators, soft robotics, dynamic models, Cosserat rod theory, Lie group variational integration, recurrent neural network

INTRODUCTION

Soft continuum manipulators are flexible and highly deformable robots composed of soft and mostly elastic materials, and can serve as possible substitutes for rigid robots. Advantages of soft manipulator robots such as their compliance, dexterity, and adaptability to complex workspaces are driving the emergent research in this field. By contrast, rigidity of traditional rigid robots limits their use in constrained and confined environments, and reduces the possibilities for safe interaction with humans. Soft continuum manipulators have found applications in many areas, such as dexterous grasping (McMahan et al., 2006; Katzschmann et al., 2015) and assistive devices (Ansari et al., 2017), and particularly in the field of minimally invasive surgeries, such as laryngeal surgery (Simaan et al., 2004), catheter-based endovascular intervention (Grady et al., 2000; Burgner et al., 2013), and cardiovascular surgery (Kesner and Howe, 2011).

Analytical modeling of soft manipulators helps evaluate their motion and determine their workspace, in order to be used for control, motion planning, and animation purposes. Soft manipulators distinguish themselves by having an infinite number of degrees of freedom in any

workspace they occupy. This characterization makes modeling complicated for soft manipulators. Several approaches have been investigated thus far in the literature. Most of the approaches consider the kinematic (i.e. static or quasi-static) modeling of the manipulators such as static analysis using virtual-work model (Xu and Simaan, 2010), Cosserat rod theory (Pai, 2002; Jones et al., 2009; Mahvash and Dupont, 2011), and α Lie group formulation (Grazioso et al., 2019). These models do not describe full dynamics of the manipulators, and they may show performance degradation when it comes to high-frequency applications or large and complex deformations. On the other hand, dynamical modeling approaches [e.g. Wen et al. (2012), Jung et al. (2014), Hyatt et al. (2019), Sadati et al. (2019), Till et al. (2019), Tariverdi et al. (2020)], contain dynamics of the manipulators and also take into account time-varying responses of manipulators, including high-frequency modes. However, the dynamic models mostly rely on traditional methods, such as finite elements and finite differences (i.e., quantitative and numerical methods), making the algorithms computationally expensive for real-time applications. In other words, to obtain sufficiently accurate solutions, methods need to deal with fine meshes, which increase memory use and computation time. Another limitation is that their solutions are discrete or not sufficiently differentiable. It is worth noting that in model-based controllers or observers, having a differentiable solution (i.e., a solution that can be evaluated continuously on the workspace) is crucial in the design process. Furthermore, when softer materials are employed for manipulator construction with more complex geometries or large deformations, modeling their behavior analytically becomes challenging. Therefore, there is a need for appropriate data-driven approaches without compromising computational bandwidths and the prediction quality.

Dynamics of soft continuum manipulators have highly nonlinear behavior and are expressed as Partial Differential Equations (PDEs). An effective approach to represent and model PDEs solutions is to use Neural Networks (NN). NN-based solutions of PDEs are infinitely differentiable by eliminating the need for interpolation. Furthermore, compared to finite elements or difference methods, solutions are represented by fewer parameters, which reduces the memory use. There are studies that use machine learning algorithms to find a solution for special types of PDEs such as (Lagaris et al., 1998; Lee and Kang, 1990; Weinan et al., 2017; Raissi et al., 2019). However, to the authors' best knowledge, there is no study that investigates possible NN-based solutions for partial differential equations that describe the full dynamics of continuum manipulators. In this work, inspired by a time-space integration scheme and by using the Lie group variational integration method (Demoures et al., 2015), the dynamic equations for translation and rotation for each node of a soft continuum manipulator are decoupled, providing an appropriate structure aimed at developing a real-time modeling algorithm. Afterward, Recurrent Neural Networks (RNNs)-based models are employed to approximate the high-dimensional discretized equations. Additionally, external torques and forces (e.g.,

control inputs, friction, and gravity) are incorporated into the model in a real-time manner for control applications.

The ability of RNNs to learn and approximate large classes of nonlinear functions over sequences of inputs accurately makes them prime candidates for use in dynamic modeling of complex nonlinear systems. RNNs with Long Short-Term Memory (LSTM) layers process sequences by iterating through the sequence elements. Using an internal feedback, the network is capable of preserving long-term dependencies. Essentially, LSTM layers prevent older information from gradually vanishing. These networks also have been used for several applications in soft robotics. To name a few, Thuruthel in Thuruthel et al. (2019) proposes a model-free, real-time sensing method for soft robots perception. The authors in Thuruthel et al. (2017) uses RNNs to model and control soft robotic manipulators. Also, force and motion estimation using RNNs has been investigated in Marban et al. (2019) and Turan et al. (2018), respectively.

This paper aims to develop a real-time dynamic model for analyzing the dynamics of soft manipulators. Investigation of previous work on the modeling of the continuum manipulators suggests that existing literature focuses primarily on static or quasi-static approaches, or does not provide a real-time model. The contribution of this article is to present a scalable, parallel and real-time modeling algorithm for soft manipulators dynamics. The contributions of this paper are as follows.

- Existing approaches primarily deal with kinematic modeling methods. Nevertheless, in this study, real-time prediction of soft manipulators full spatial dynamics is considered in the proposed RNN-based algorithm by proposing multiple light-weight RNN-based models.
- In traditional modeling approaches, there are no systematic methods to obtain knowledge about dissipation forces, in particular friction, in the modeling procedure. The presented algorithm intrinsically takes the dissipation forces into account and incorporates their effects into the model.
- Through an experiment, results of the proposed RNN-based model and Cosserat rod theory method are compared, revealing the practical effectiveness of the proposed methodology.

The remainder of this paper is organized as follows: the problem statement is given in **Section 2**. **Section 3** describes the proposed RNN-based algorithm in details. In **Section 4 and Section 5**, different simulations and experimental validation are presented to demonstrate the efficacy of the proposed RNN-based method, in terms of the model performances and accurately predicting poses of manipulators. Finally, the main conclusions are stated in **Section 7**.

2 PROBLEM STATEMENT

Consider a continuum manipulator with large deflections described by dynamic equations of motion [as presented in

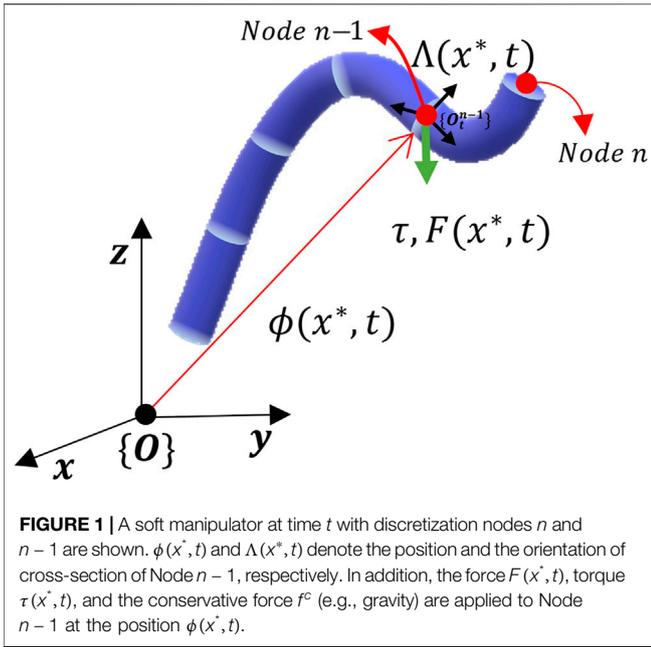


FIGURE 1 | A soft manipulator at time t with discretization nodes n and $n - 1$ are shown. $\phi(x^*, t)$ and $\Lambda(x^*, t)$ denote the position and the orientation of cross-section of Node $n - 1$, respectively. In addition, the force $F(x^*, t)$, torque $\tau(x^*, t)$, and the conservative force f^c (e.g., gravity) are applied to Node $n - 1$ at the position $\phi(x^*, t)$.

Tariverdi et al. (2020) and Demoures et al. (2015)] in the PDEs form as

$$H\omega_t + \omega \times H\omega + n \times \Lambda^{-1}\phi_x - \Lambda^{-1}\Lambda_x \times m - m_x = \Lambda^{-1}\tau$$

$$M\phi_{tt} - \Lambda(\Lambda^{-1}\Lambda_x \times n) - \Lambda n_x + f^c = f \quad (1)$$

where $M = \rho \times A$ (ρ and A are the manipulator constant mass density and its cross-section area), $\omega \in \mathbb{R}^3$ is the manipulator's angular velocity, $H \in \mathbb{R}^{3 \times 3}$ is the manipulator's inertia matrix, $\phi \in \mathbb{R}^3$ is the position of the manipulator's line of centroids in its workspace, $\Lambda \in SO(3)$ denotes the orientation of moving cross-sections at point ϕ . Also, $n \in \mathbb{R}^3$ and $m \in \mathbb{R}^3$ are the stresses and momenta along the manipulator, $f^c \in \mathbb{R}^3$ represents conservative forces (e.g. gravity). Furthermore, $(\cdot)_x$, $(\cdot)_t$, and $(\cdot)_{tt}$ denote partial derivatives with respect to position, time, and the second partial derivative with respect to time, respectively. Finally, $f \in \mathbb{R}^3$ and $\tau \in \mathbb{R}^3$ are non-conservative forces and torques (e.g., frictions and control inputs)¹.

Although high fidelity models given in the references can describe continuum manipulators dynamics efficiently, they suffer from limitations that are discussed in **Section 1**. Inspired by the structure and formulation of the dynamics based on the Lie group variational integration scheme, the aim is to propose distributed deep recurrent neural networks to capture and simulate soft manipulators dynamics in real-time to be able control them more accurately than existing models.

3 PROPOSED RNN-BASED MODEL

This section is devoted to develop a model based on the time series prediction using RNNs. To solve PDEs numerically using

NNs, one approach is to utilize discrete solutions of finite element or difference methods to train an NN. A Lie group variational time integration model is employed to discretize the continuous dynamics of a soft manipulator². The whole manipulator is discretized into an arbitrary number of nodes where the position and orientation equations of each node are decoupled. In our study, we discretize the manipulator with equidistant nodes, but this can be changed depending on the application.

Figure 1 demonstrates a soft continuum manipulator at time t where x^* is the undeformed length of Node $n - 1$. The force $F(x^*, t)$, torque $\tau(x^*, t)$ are applied to Node $n - 1$ at the position $\phi(x^*, t)$. Also, $\Lambda(x^*, t)$ is the orientation matrix from the frame $\{O\}$ to the frame $\{O_t^{n-1}\}$ attached to the cross-section of Node $n - 1$.

The discrete equations suggest an appropriate structure for the RNNs-based model. Given time-sequence inputs (as a first input layer), i.e., poses (positions and orientations) of nodes, and also forces and torques (as a second input layer) applied to each node, the RNN-based model of Node n is depicted in **Figure 2A,B**. For Node n , the first input layer is a time-sequence series of poses p_{n-1} , p_n , and p_{n+1} (i.e., poses of Node n and its adjacent nodes $n - 1$ and $n + 1$) and the second input layer includes forces and torques of node n at time t , i.e., $[F_t^n, \tau_t^n]^T$ which are incorporated into the model through dense layers.

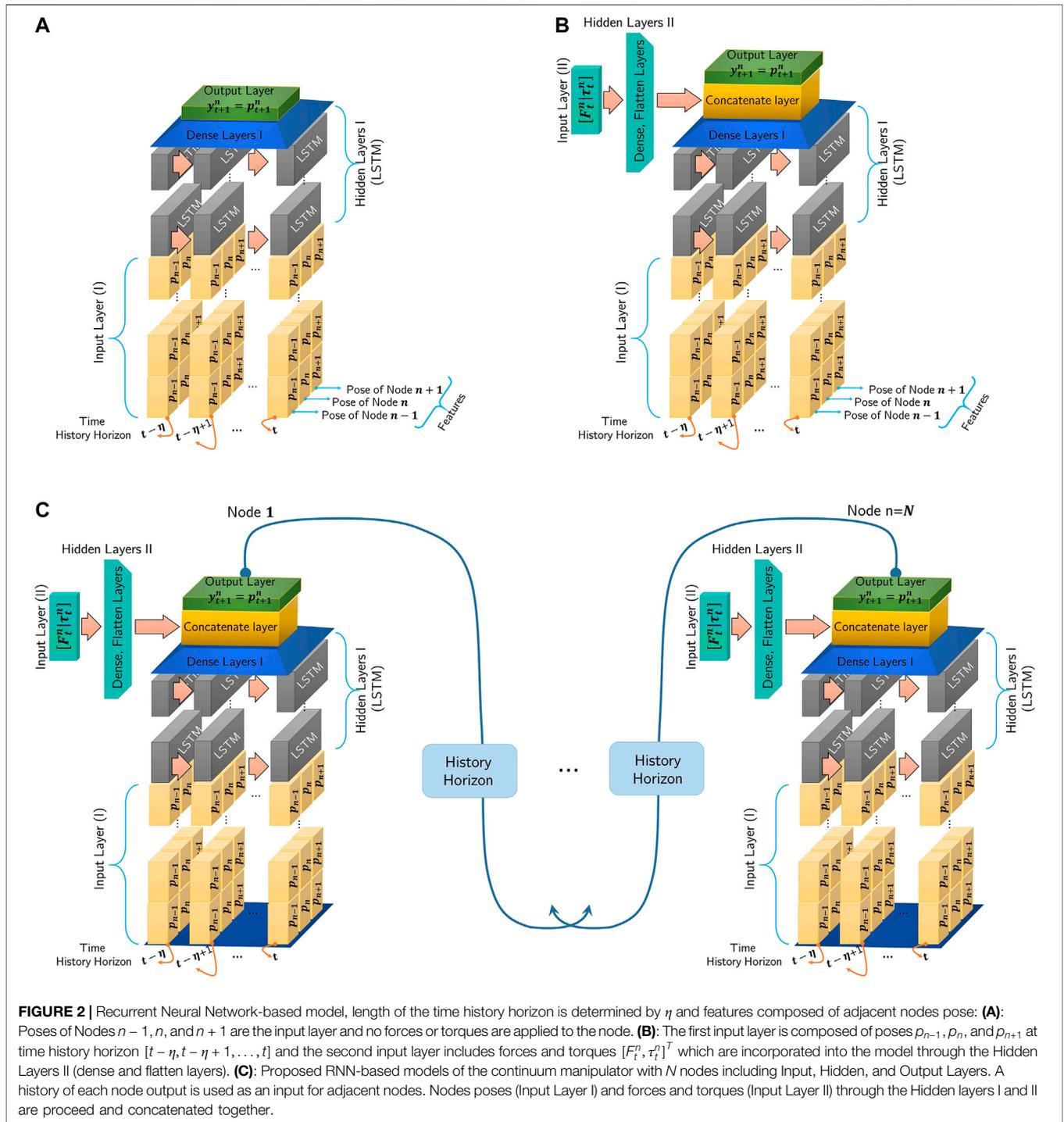
The network takes specific size vectors as inputs, which are called input layers. The inputs are transformed through a series of hidden layers (LSTM, dense, or fully connected layers) to produce an output. The output vectors are called an output layer. Dense or fully connected layers perform linear operations (i.e., multiplication and summation) on their inputs. Furthermore, LSTM layers consist of LSTM units, which can process sequences of data of any length, for example, poses of nodes. An LSTM unit controls contributions of each element of the input layer in the output and keeps track of the dependencies between the elements (Hochreiter and Schmidhuber, 1997).

For the training process, data-sets contain time-sequence inputs and forces and torques applied to each node. Also, for each node, the poses of the node and its neighbors are considered features, as shown in **Figure 2A,B**. The first and second input layers proceed through LSTM layers and dense layers as hidden layers, respectively. Finally, output layers have resulted from fully connected layers.

By augmenting the given models for all nodes (see **Figure 2B**) as a series, the proposed RNN-based models of the whole continuum manipulator with N nodes with non-conservative forces and torques are depicted in **Figure 2C**. Output of every node is updated at each time step by using a history (at least two previous time steps) of neighboring node outputs. Therefore, the proposed architecture suggests a suitable framework to construct a parallel modeling algorithm.

¹For the details see Demoures et al. (2015).

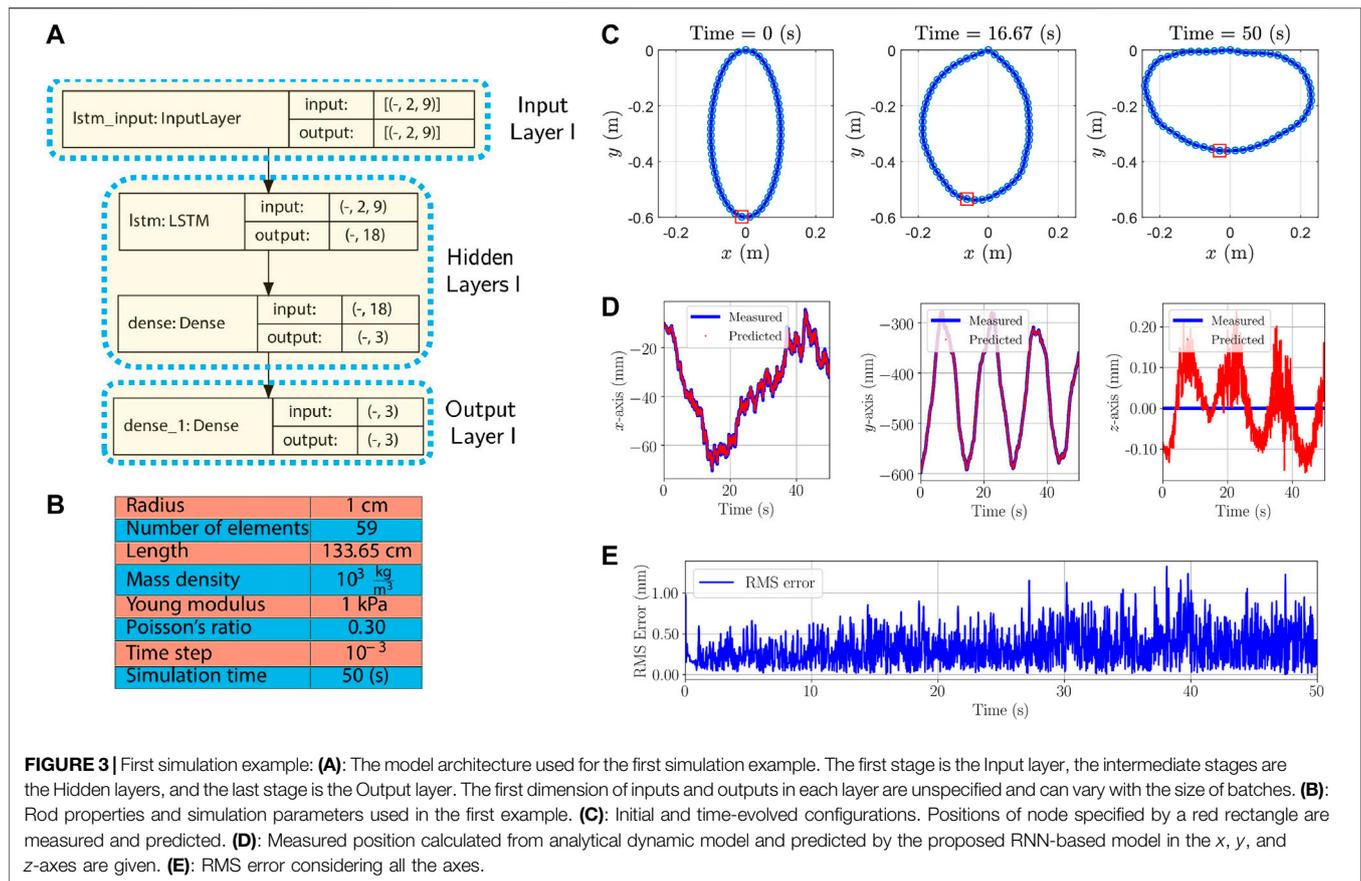
²Dynamic equations are given in Tariverdi et al. (2020), Sec. 2.



4 SIMULATIONS RESULTS

In this section, we consider different examples and evaluate the performance of the proposed RNN-based model in **Figure 2C**. It is worth mentioning that data-sets play a crucial role in efficiency and accuracy in machine learning-based algorithms. The data acquisition process from a robot in real-world environments is both time and cost-consuming (implementation of multiple

sensors, data filtering, and fusion, etc.). As an alternative approach, the required data can be acquired through simulations of high fidelity models. The obtained data can thus be transferred to train the algorithms to be implemented in real-world scenarios. In this section and for the presented examples, required data for the training of the proposed RNN-based model are acquired through simulations of the algorithm presented in Tariverdi et al. (2020), Sec. 2). For clarity, this model



is henceforth referred to as the analytical dynamic model. In addition, since thin rods are considered in the examples, orientations of cross-sections are not of any concern. Also, it should be noted that orientations, except the twisting angle, can be reconstructed from manipulators' configuration. Therefore, to obtain a computationally light model, the focus of attention is only on the prediction of positions.

4.1 First Simulation: An Ellipse Without External Wrenches

As a first case, a cylindrical rod is bent into a circle and its ends are attached to one another. The rod is then deformed into an elliptical shape and released. Due to potential energies in the ellipse, it starts to move without any external disturbances. The goal is to model the behavior of the ellipse resultant from its internal elastic energy.

The ellipse is formed in the xy -plane with the width 0.2 m and height 0.6 m. As boundary conditions, the first and last nodes are fixed to the origin and their orientations are set to $R_y(18.07^\circ)$ and $R_y(341.92^\circ)$, respectively, where $R_y(\theta)$ denotes a rotation matrix describing a rotation around the y -axis by θ degrees. The rod properties, simulation parameters, and the structure of the proposed RNN-based model are given in **Figure 3A,B**. Furthermore, the initial and a few time-evolved configurations are shown in **Figure 3C**. As seen, the ellipse oscillates back and

forth due to its internal elastic energy. Since orientations except the twisting angle can be reconstructed from the configuration of the manipulator, to have a light model and for brevity, positions of the node located at $(-0.01, -0.59)$ —Node 30th—are predicted. The chosen node is the furthest from the origin and would, compared to other nodes, most likely have the largest errors.

50,001 position samples are generated from the analytical model for each node. We augment 1-by-3 position vectors of Node 30th and its adjacent nodes (Nodes 29th and 31st) at each time step. Therefore, the augmentation results in a 1-by-9 vector. Furthermore, the size of history horizons is chosen to be 2. In other words, $\eta = 1$ in **Figure 2A,B**. Finally, augmented 2-by-9 tensors are obtained for each time step. The prepared data-set is called Data-set I and 60 percent of it is used for training process. The architecture in **Figure 3A** shows the input layer consists of tensors of size 2×9 . The first dimension of all layers are reserved for batch sizes and for the training, the batch size 1 was chosen. In the architecture of the model in **Figure 3A**, the Input, Hidden and Output Layers I together with the number of nodes and type of layers are demonstrated according to **Figure 2A**.

First, we evaluate the model by using unseen data samples in Data-set I and the results are shown in **Figure 3D**. The maximum and mean absolute error are (1.57 mm, 0.27 mm), (2.27 mm, 0.46 mm), and (0.23 mm, 0.06 mm), or in other words, the percentage of the maximum errors with respect to the length of the manipulator are 0.11, 0.17, and 0.02 in the x , y , and z -axes, respectively. It is worth

mentioning that it is prior knowledge that the manipulator does not have any motion in the z -axis and therefore, components of the z -axis in position vectors can be ignored. Furthermore, the evaluation Root-Mean-Square (RMS) errors of the considered node in all axes at every time step is calculated by

$$RMSE(t) = \sqrt{\frac{1}{3} \left((x_p(t) - x_m(t))^2 + (y_p(t) - y_m(t))^2 + (z_p(t) - z_m(t))^2 \right)} \quad (2)$$

where predicted positions $[x_p(t), y_p(t), z_p(t)]^T$ obtained from the proposed RNN-based model and measurement positions $[x_m(t), y_m(t), z_m(t)]^T$ in Data-set I and the results are shown in **Figure 3E**.

To demonstrate that the model can be extended to different boundary and initial conditions, the cylindrical rod is employed to form a horizontal ellipse with the width 0.6 m and height 0.2 m. The rod properties and the simulation parameters given in **Figure 3B** are used. As boundary conditions, the first and last nodes are attached to the origin and their orientations are set to the identity. The manipulator with the new boundary and initial conditions is only used for the evaluation of the trained model by predicting positions of the node located at $(-0.01, -0.19)$. Based on the prediction, the maximum and mean absolute errors are (26.33 mm, 3.37 mm), (21.71 mm, 3.70 mm), and (4.92 mm, 4.22 mm) in the x , y , and z -axes, respectively. Furthermore, the maximum/worst-case errors with respect to the length of the manipulator are 1.97%, 1.62%, and 0.37% in each axis, respectively.

Let us assume that the analytical dynamic model is implemented in a parallel scheme, i.e., each node of 59 nodes is handled with a CPU core or different hardware such that there is no latency in communications. Then, the dynamics of each node can be solved in 1.62×10^{-4} s on average. In addition, to preserve the convergence of the solver of the analytical dynamic model, the maximum constant time step for this simulation is 10^{-3} s. A minimum criterion to have a real-time performance is that the time required to solve each node dynamics must be less than the constant step simulation. To be more specific, to have a real-time model, the CPU time, i.e., the amount of time spent in a user code must be less than Wall-clock time that measures the time elapsed to run a user code. According to this minimal criterion, as long as the computation-time for simulation of a method/model is less than a user-defined time for the simulation, the model is called a real-time model. It can be shown that in this example and based on the given assumption, the maximum bandwidth for a real-time performance is 3.93 Hz on average (calculation is done on a 16 GB, 1.99 GHz Intel i7 machine running windows 10). It should be pointed out that we use the same machine for calculations in this paper. It will be discussed that even achieving this bandwidth limit is not feasible. On the other hand, for the proposed RNN-based model, the bandwidth of a real-time performance is 65.70 Hz, which can be further improved by optimizing the number of layers and trainable parameters.

It is worth mentioning that the considered assumption is very strict, which cannot be satisfied in reality. First of all, conventional algorithms need a relatively high number of nodes to have numerical stability and an acceptable convergence rate. Furthermore, due to limitations in computation resources, more than one node will be assigned

to each core of CPU, and there is always latency in communications between threads in parallel programmings. Therefore, reaching the mentioned bandwidth through the analytical dynamic model is infeasible. However, the real-time performance of the proposed RNN-based model can be applicable in closed-loop control applications.

4.2 Second Simulation: A Cylindrical Rod With External Wrenches

In the second example, we simulate a rod with a circular cross section, which is actuated by external forces such that its tip tracks a square in space. In this example, the goal is to model the behavior of the rod which results from applied external forces on its end-effector. For boundary conditions, the first node is fixed to the origin and its orientation is set to the identity for all time steps. The rod properties and simulation parameters, and the structure of the proposed model are given in **Figure 4**. The trajectory of the end-effector and the applied forces onto it are shown in **Figure 5A**.

160,001 position and force samples are generated from the analytical model for each node. We augment 1-by-3 position vectors of the last node (end-effector) and its adjacent node at each time step. Therefore, the augmentation results in a 1-by-6 vector. Furthermore, the size of history horizons is chosen to be 2 ($\eta = 1$). Finally, augmented 2-by-6 tensors are obtained for each time steps which are fed to the model as the Input Layer I. The same preparation process are applied for the force data samples which are used as the Input Layer II. The prepared data-set is called Data-set II and 60% of it is used for training process. The architecture in **Figure 4A** shows the Input layers I and II consist of tensors of size $(Batch\ Size \times 2 \times 6)$ and $(Batch\ Size \times 2 \times 3)$, respectively. The first dimension of all layers are reserved for batch sizes and for the training, the batch size 1 was chosen. In the architecture of the model in **Figure 4A**, the Input, Hidden and Output Layers I and II together with the number of nodes and type of layers are demonstrated according to **Figure 2B**.

First, unseen data samples in Data-set II are employed to evaluate the model, and tip positions are calculated and the results are shown in **Figure 5B**. The maximum and mean absolute error are (3.58 mm, 1.70 mm), (1.80 mm, 0.69 mm), and (2.73 mm, 1.41 mm), or in other words, the maximum errors with respect to the length of the manipulator are 0.71%, 0.36%, and 0.54% in x , y , and z -axes, respectively. The RMS errors of the end-effector through **Eq. 2** are shown in **Figure 5C**.

To evaluate the generalizability of the trained model, different profiles of forces are applied to the model aiming at obtaining different position trajectories for the end-effector as depicted in **Figure 6A,D**. To fulfill the goal of the second example, the new forces are only used for the evaluation of the trained model by predicting the positions of the end-effector. Results of the prediction are plotted in **Figure 6B** and are as follows: The maximum and mean absolute errors are (10.49 mm, 2.61 mm), (5.54 mm, 1.05 mm), and (5.97 mm, 2.83 mm), furthermore, the percentage of the maximum/worst-case errors with respect to the length of the manipulator are 1.90, 1, and 1.08 in the x , y , and z -axes, respectively. The RMS errors of the end-effector through **Eq. 2** are shown in **Figure 6C**.

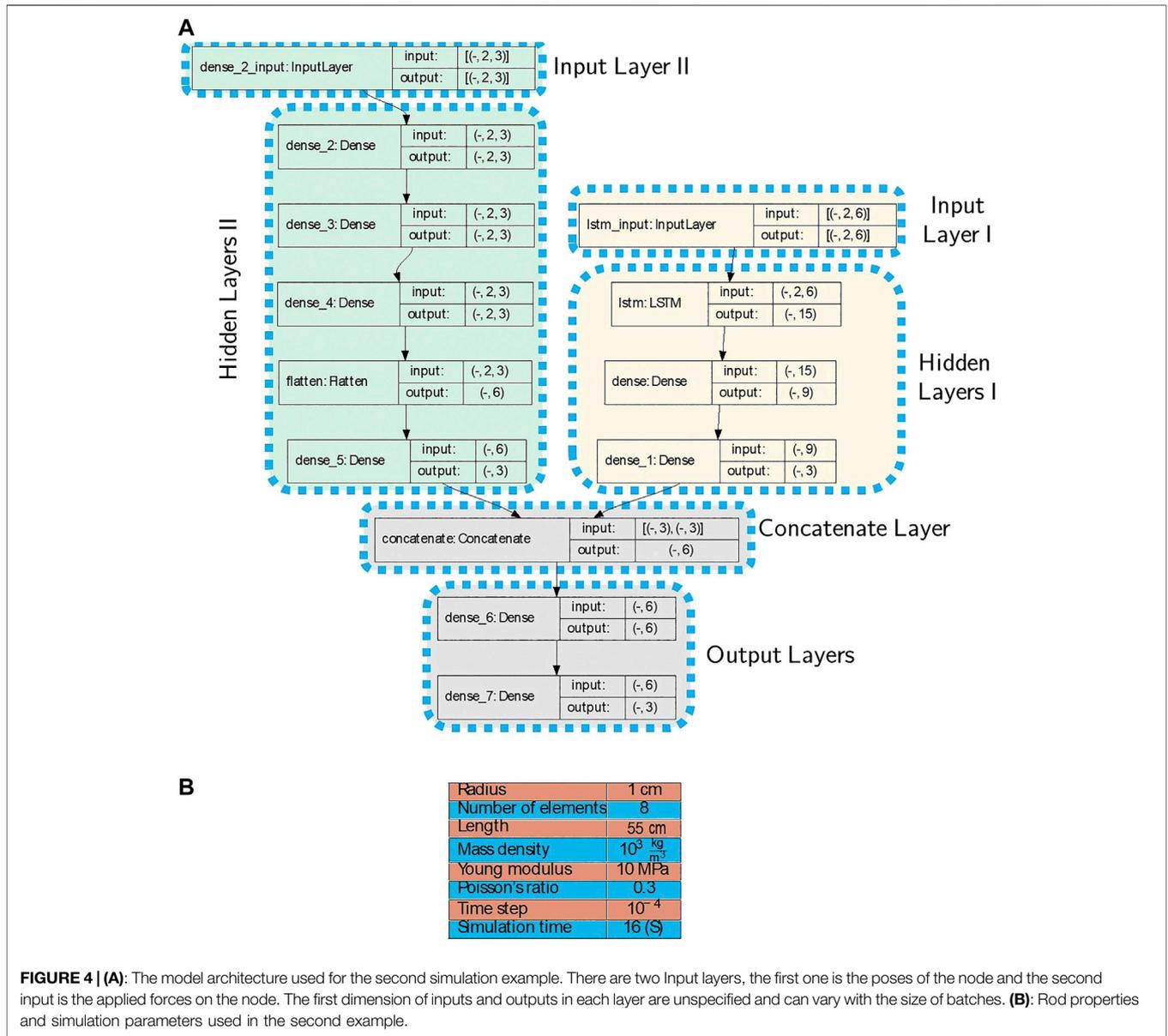


FIGURE 4 | (A): The model architecture used for the second simulation example. There are two Input layers, the first one is the poses of the node and the second input is the applied forces on the node. The first dimension of inputs and outputs in each layer are unspecified and can vary with the size of batches. **(B):** Rod properties and simulation parameters used in the second example.

In this example, the maximum constant time step for this simulation is 10^{-4} s, to have a convergent numerical solver for the analytical dynamic model. In addition, on average, the time 1.89×10^{-4} s is required for solving the dynamics of each node. In other words, the analytical dynamic model can not achieve any real-time performance for this example. However, the proposed model achieves a real-time performance of the bandwidth of 60.30 Hz on average.

4.3 Third Simulation: A Cylindrical Rod With and Without External Wrenches

In the last example, we form a semi-circular shape with a cylindrical rod. A force is applied to the middle node—Node 51th—in the $-y$ -axis direction for 0.5 s and then the force is removed. Furthermore, the boundary conditions are as follows:

the first and last nodes are fixed to the origin and their orientations are set to the identity and $R_z(181.81^\circ)$, respectively, where $R_z(\theta)$ describes rotation around the z -axis by θ degrees. In this example, the idea is to model the behavior of the rod resulted from applied external forces and internal elastic energy. The structure of the proposed model is given in **Figure 7A** and the rod properties and simulation parameters are given in **Figure 7B**. The initial and a few time-evolved configurations together with the applied forces are given in **Figure 8A**.

20,001 position and force samples are generated from the analytical model for each node. We augment 1-by-3 position vectors of Node 15th and its adjacent nodes at each time step. Therefore, the augmentation results in a 1-by-9 vector. Furthermore, the size of history horizons is chosen to be 2 ($\eta = 1$). Finally, augmented 2-by-9 tensors are obtained for

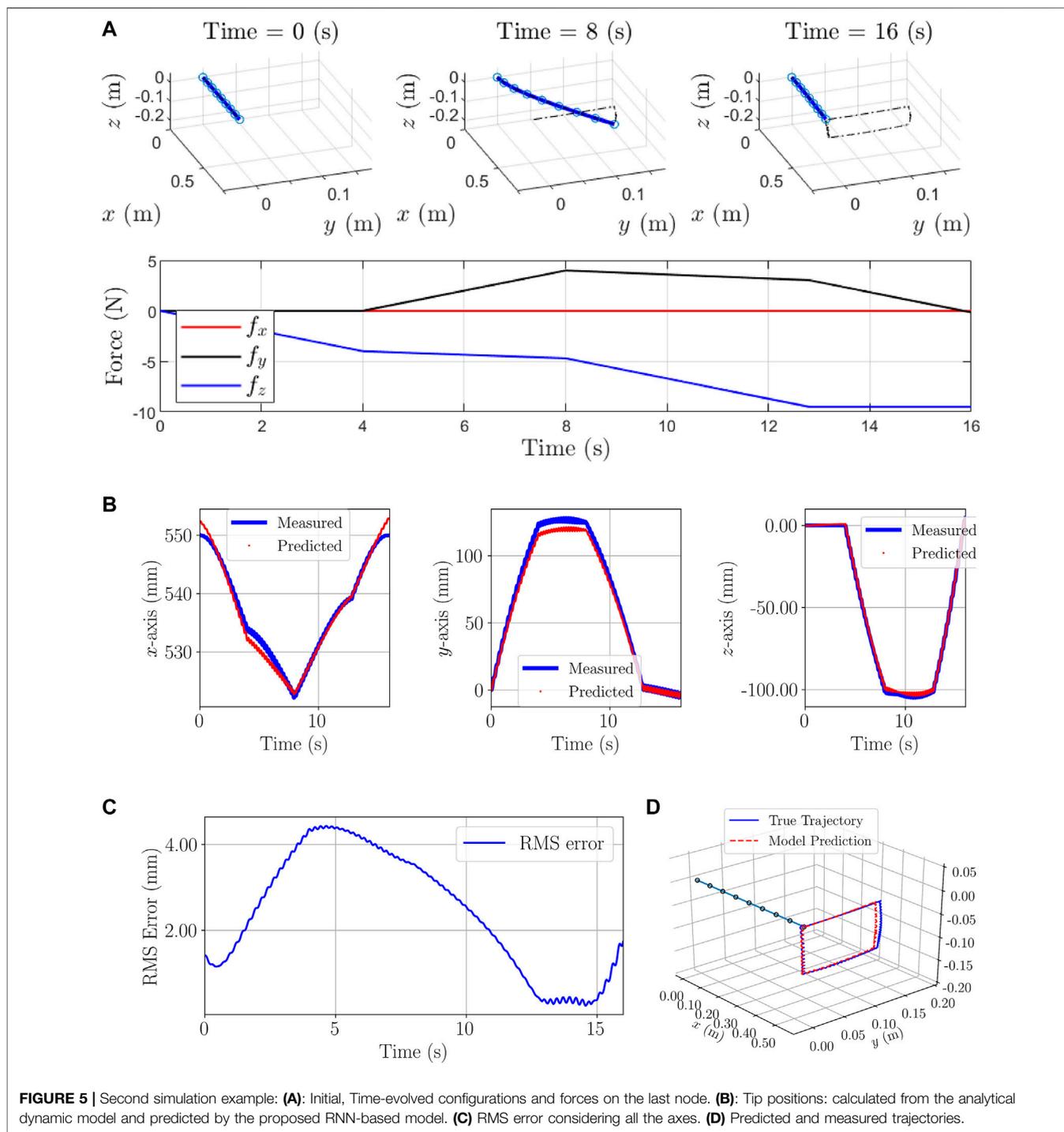
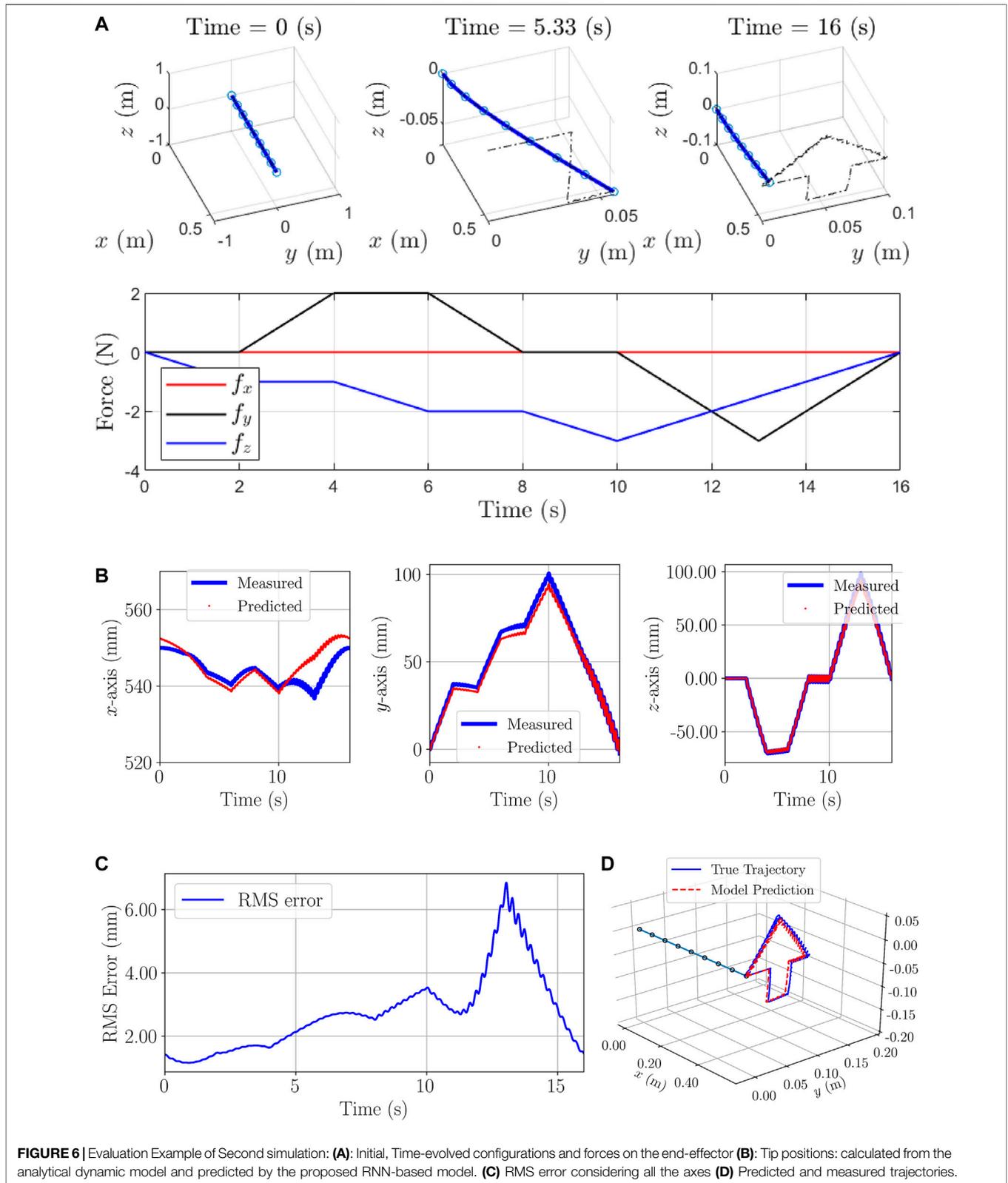


FIGURE 5 | Second simulation example: **(A):** Initial, Time-evolved configurations and forces on the last node. **(B):** Tip positions: calculated from the analytical dynamic model and predicted by the proposed RNN-based model. **(C)** RMS error considering all the axes. **(D)** Predicted and measured trajectories.

each time steps which are fed to the model as the Input Layer I. The same preparation process are applied for the force data samples which are used as the Input Layer II. The prepared dataset is called Data-set III and 60% of the data is used for training process. The architecture in **Figure 7A** shows the Input layers I and II consist of tensors of size $(Batch\ Size \times 2 \times 9)$ and $(Batch\ Size \times 2 \times 3)$, respectively. The first dimension of all

layers are reserved for batch sizes and for the training, the batch size 1 was chosen. In the architecture of the model in **Figure 7A**, the Input, Hidden and Output Layers I and II together with the number of nodes and type of layers are demonstrated according to **Figure 2B**.

The positions of Node 51th are predicted using seen and unseen data samples in Data-set III and the results are shown in



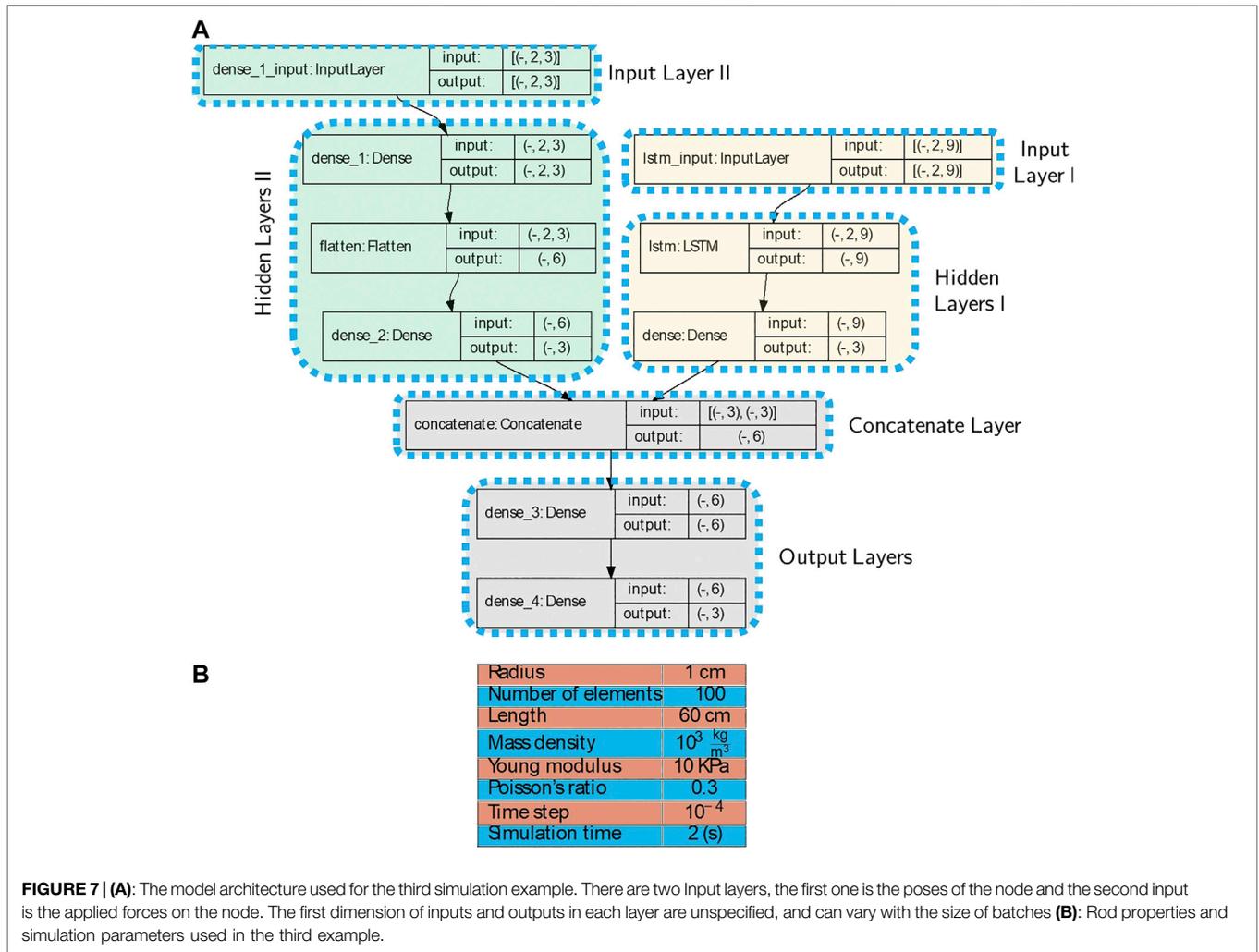


Figure 8B,C. The maximum and mean absolute error are (1.36 mm, 0.78 mm), (0.23 mm, 0.13 mm), and (2.22 mm, 0.81 mm). Furthermore, the maximum/worst-case errors with respect to the length of the manipulator are 0.22%, 0.04%, and 0.37% in the x , y , and z -axes, respectively.

For the evaluation of the trained model and to fulfill the goal of this example, force vector $[0, 0, -100 \times \cos(\pi t)]^T$ mN mN is applied to Node 51th for $t \in [1, 2]$ s. Results of the prediction are as follows: The maximum and mean absolute errors are (3.73 mm, 1.25 mm), (2 mm, 0.2 mm), and (8.1 mm, 1.2 mm), furthermore, the maximum/worst-case errors with respect to the length of the manipulator are 0.62%, 0.33%, and 1.35% in the x , y , and z -axes, respectively.

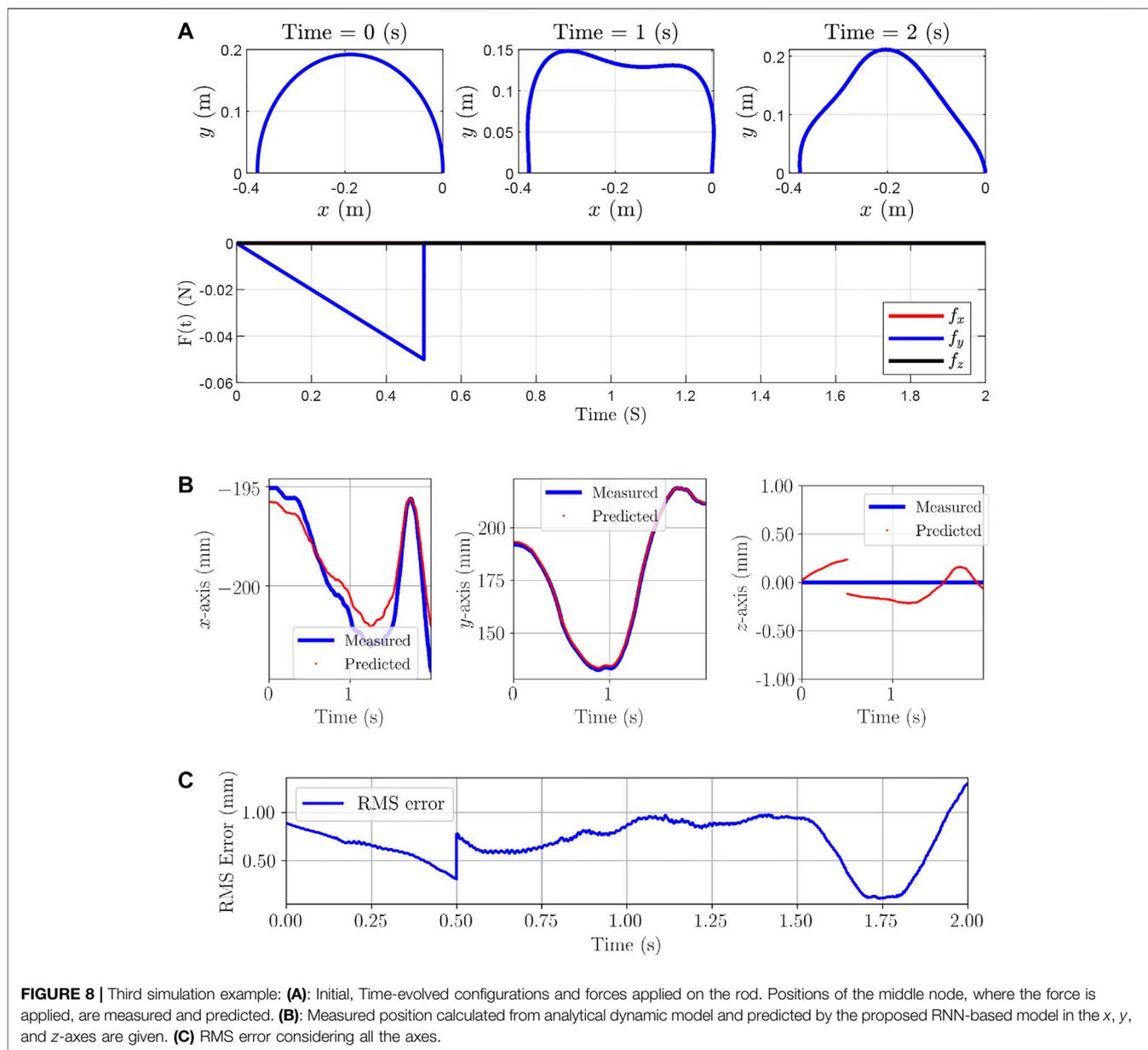
In this example, the maximum constant time step for this simulation is 10^{-4} s using the analytical model. In other words, the analytical model does not show a real-time performance since, on average, the time 2.22×10^{-4} s is required for solving the dynamics of each node. On the other hand, the proposed model can achieve a real-time performance of the bandwidth 58.13 Hz on average.

5 EXPERIMENTAL RESULTS

This section is devoted to the experimental validation of the presented model. To that end, we fabricated a soft manipulator on which magnetic fields are used to produce necessary forces and torques. Compared to the simulations in which positions are predicted, time-sequence input is composed of orientations of nodes in the experiment. Furthermore, to show the performance of the algorithm, results from the presented method and a Cosserat rod-based theoretical model are compared to show the efficiency of the proposed RNN-based model. The Cosserat rod model of the soft manipulator is detailed in Appendix.

5.1 Soft Continuum Manipulator

A soft continuum manipulator is fabricated from a urethane rubber Polymer Matrix Composite 770 (PMC-770, Smooth-On Inc., Macungie, United States) and neodymium (NdFeB) block magnets whose dimensions are given in **Figure 9A**. When the manipulator is subjected to an external magnetic field, the



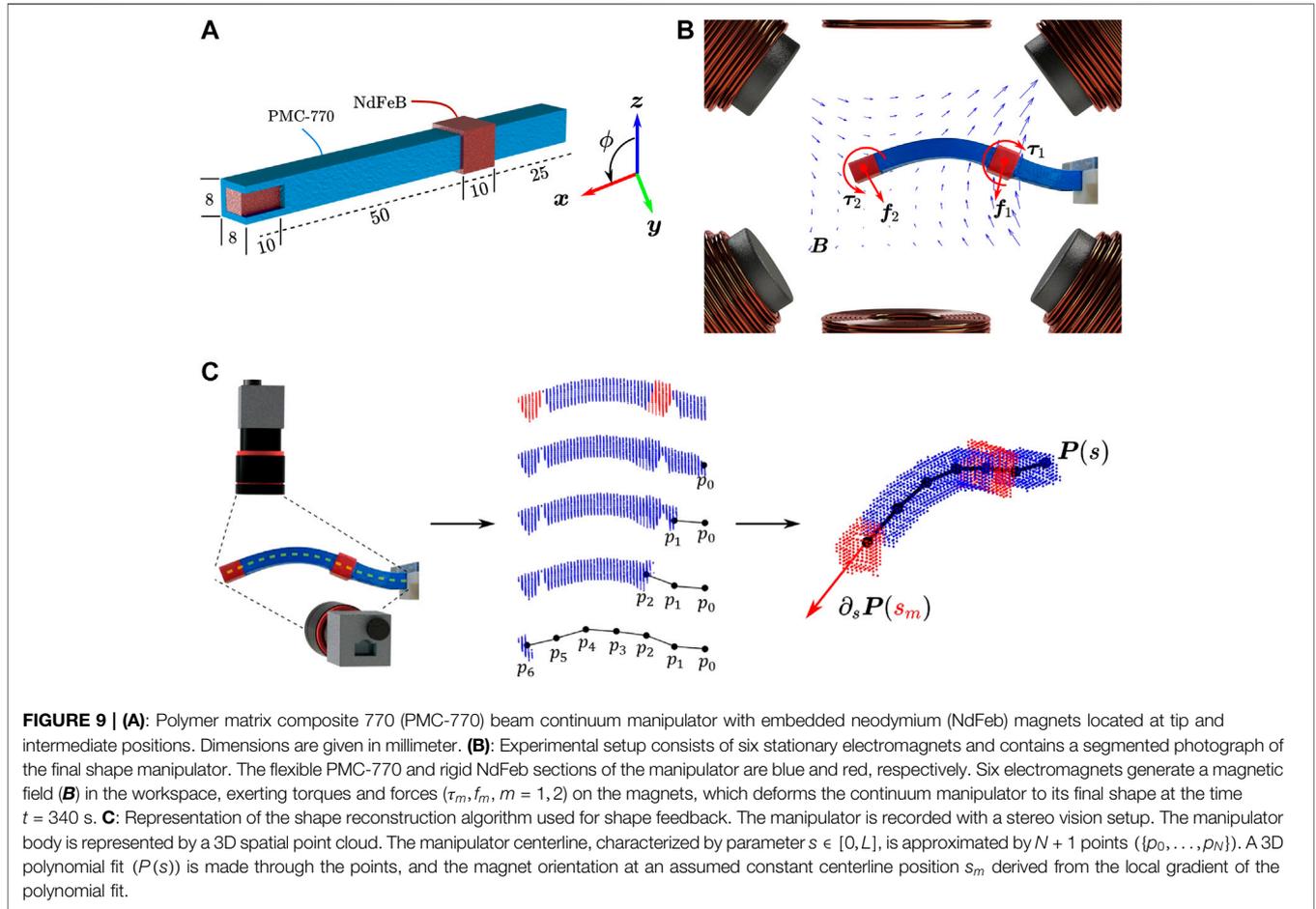
embedded magnets experience forces and torques. This causes the flexible portions of the manipulator comprised of the PMC to undergo elastic deformation.

The PMC-770 has a density $\rho = 1000 \text{ kg/m}^3$, Young modulus $E = 2.5 \text{ MPa}$, and Poisson ratio $\nu = 0.5$. The distal and proximal NdFeb magnets have grades N45 and N42, respectively. In addition, they have density $\rho = 7000 \text{ kg/m}^3$, Young modulus $E = 41.4 \text{ GPa}$, and Poisson ratio $\nu = 0.3$. It should be pointed out that Young's modulus and densities of the soft manipulator constituent materials were determined using a combination of supplier data and experiments until theoretical results (predicted by the Cosserat rod model) would resemble the experiment results. The magnitude of the magnetic dipoles carried by the manipulator was calculated from the magnets volume and manufacturer-supplied residual flux density.

5.2 Experimental Setup

The experimental setup consists of 6 stationary electromagnets surrounding a spherical workspace of 100 mm diameter Sikorski et al. (2017). **Figure 9B** shows the setup of the experiment. In addition, the final shape of manipulator has been segmented and is shown in the workspace. The continuum manipulator is suspended horizontally (along x) in the workspace and actuated to move in a plane, steering the magnets by manipulating the magnetic field generated by the electromagnets. Orientations are represented using the axis-angle notation. Let $k_m \in \mathbb{R}^3$ and $\phi_m \in \mathbb{R}$ denote the axis- and angle-of-rotation, respectively, where $m = 1, 2$ denotes the magnet index counting from the manipulator base. In the 2D experiment, $k_m = y$, and ϕ_m is defined relative to z .

Figure 9C represents the shape reconstruction of the soft manipulator through images coming from two Dalsa Genie



Nano C1940 Red-Green-Blue (RGB) cameras (TeledyneDalsa, Waterloo, ON, Canada). The flexible PMC-770 and rigid NdFeB magnets were colored blue and red, respectively. The RGB cameras (horizontal and vertical) that formed a stereo vision setup recorded the workspace during experiments. First, we discretize the actuation workspace into voxels. The silhouette of the continuum manipulator is segmented as binary masks and the manipulator body represented as a 3D spatial point cloud. The manipulator centerline is approximated by $N \in \mathbb{N}$ discrete segments. A simple iterative shape reconstruction algorithm Sikorski et al. (2019) moves through the voxels to represent the manipulator centerline with N discrete points ($\{p_0, p_1, \dots, p_N\}$) as a function of centerline parameter $s \in [0, L]$. To be specific, with knowledge of the RGB-camera frames, the points are projected onto each camera image. If a point is projected onto both binary masks, the point falls within the manipulator. This process is repeated for all voxels. Subsequently a 3D polynomial fit ($P(s)$) is made through the points. We assume that magnetically exerted forces and torques are insufficient for the manipulator extension along the centerline, and therefore assume constant positions of the magnets along the centerline, $s_m \in (0, L)$. The measured magnet position is thereafter obtained from the polynomial fit ($p_m = P(s_m)$), and its orientation from the local gradient of the polynomial fit ($\partial_s P(s_m)$) relative to a reference z axis,

$$\phi_m = \cos^{-1}(z \cdot [\partial_s P(s_m)]^\wedge)$$

$$k_m = [z \times \partial_s P(s_m)]^\wedge$$

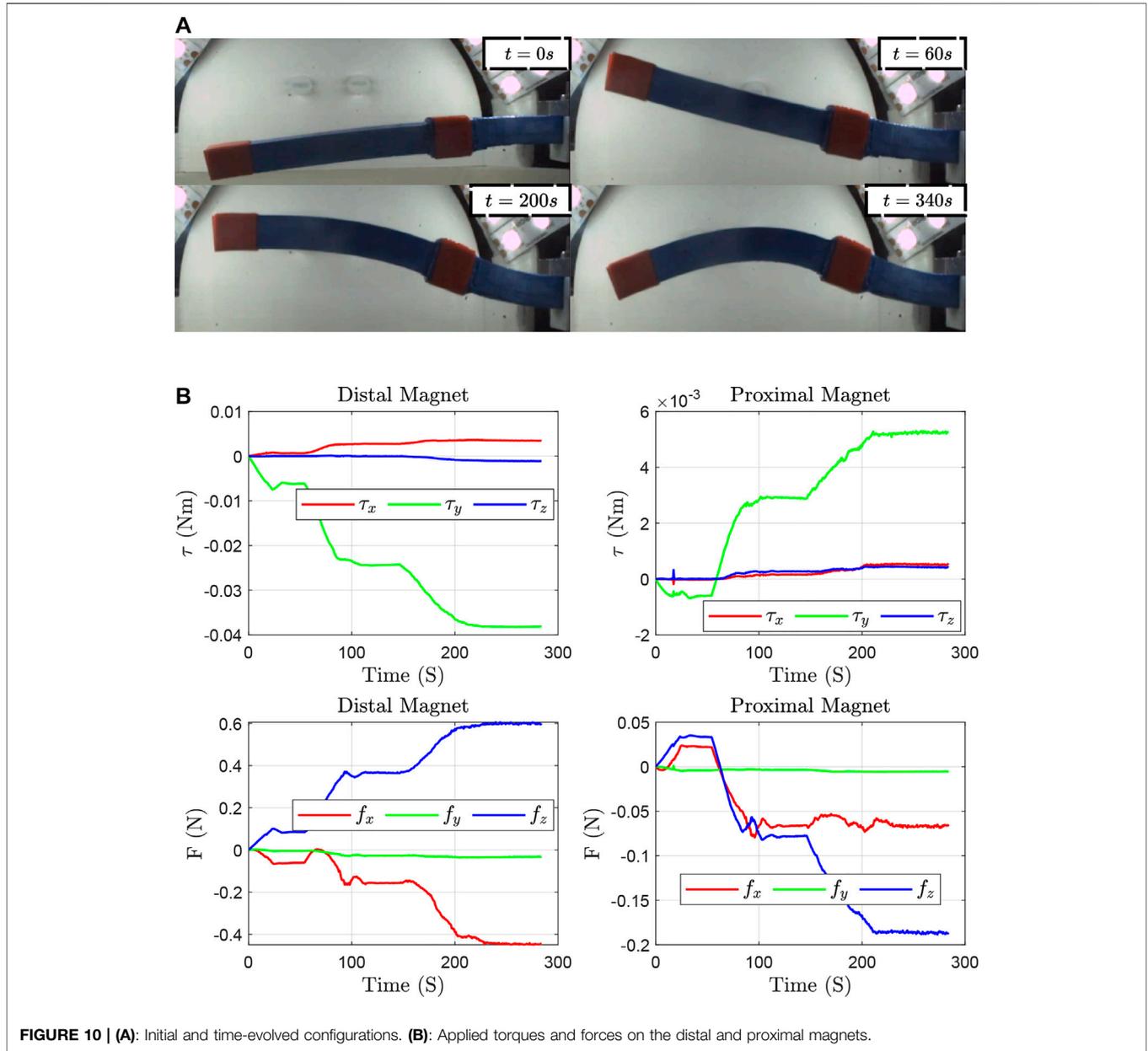
where $[\cdot]^\wedge$ represents a normalization. Furthermore, in the experiments performed for this study, camera occlusions did not occur.

The magnetic torques and forces were computed from the magnets position p_m , magnets dipole moments $\mu_m \in \mathbb{R}^3$, and electromagnet currents $I_C \in \mathbb{R}^6$. The magnets position and orientations were obtained from the stereo vision setup. Afterwards, the orientations are used to compute magnets dipoles. To compute the magnetic field, each electromagnet is associated with a unit-current field and field gradient map ($\beta_i(p) \in \mathbb{R}^3$ and $\beta_{\nabla,i}(p) \in \mathbb{R}^{3 \times 3}$, $i = 1, \dots, 6$), which computes the unit-current contribution of the electromagnet to the field at field point p . We define a map $\mathcal{G}(\beta_{\nabla,i}) : \mathbb{R}^{3 \times 3} \rightarrow \mathbb{R}^5$ which takes the five independent gradient terms of the field (Petruska and Nelson, 2015). The field (gradient) at magnet position p_m is then given by the superposition principle

$$\mathbf{B}_m = [\beta_1(p_m), \dots, \beta_6(p_m)] I_C$$

$$\mathcal{G}(\mathbf{B}_{\nabla,m}) = [\mathcal{G}(\beta_{\nabla,1}(p_m)), \dots, \mathcal{G}(\beta_{\nabla,6}(p_m))] I_C.$$

The torques and forces exerted on the magnets due to the field is given by



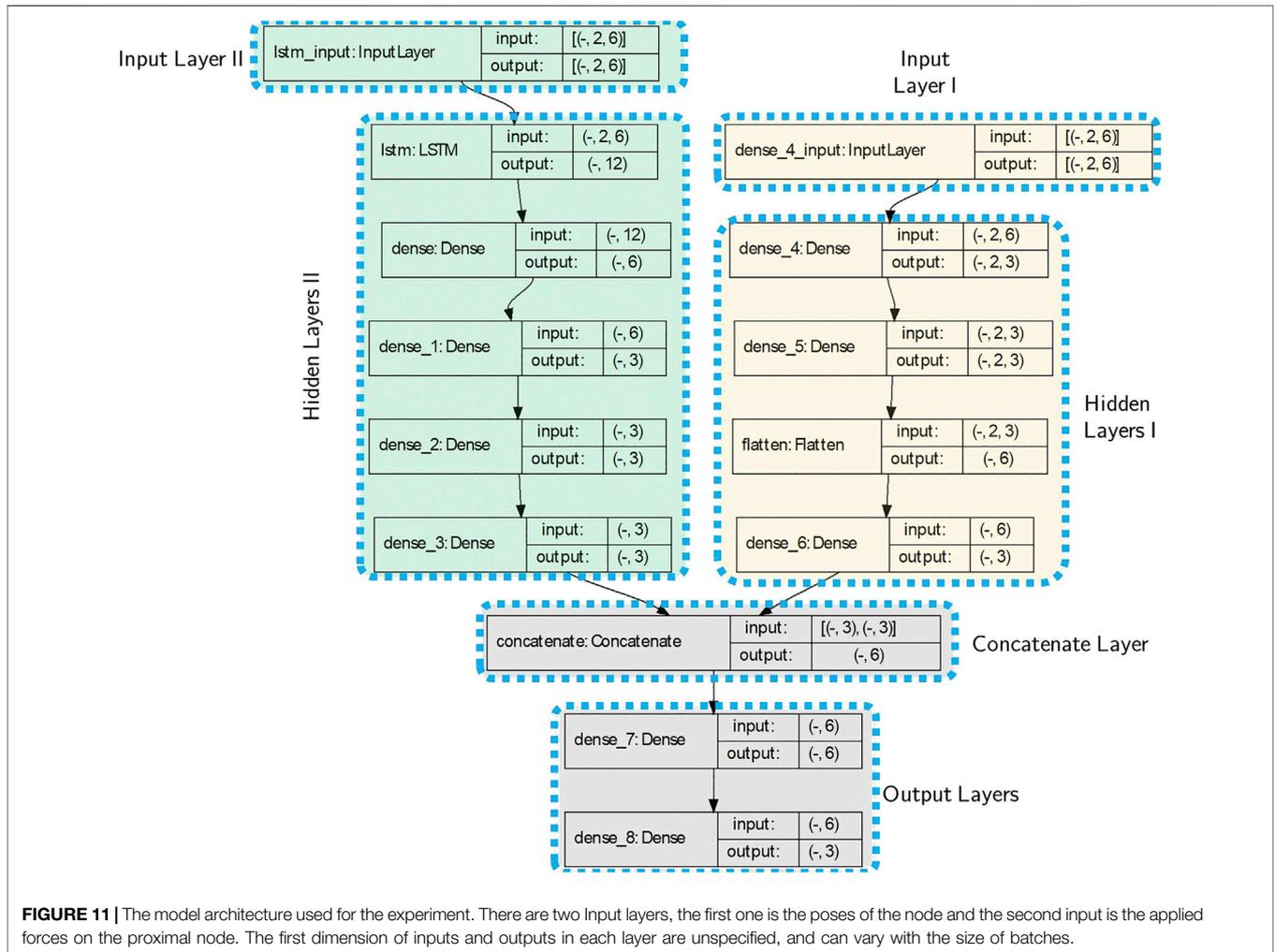
$$\begin{aligned} \tau_m &= \mu_m \times \mathbf{B}_m \Rightarrow \tau_m = [\mu_m]_{\times} \mathbf{B}_m \\ f_m &= \nabla(\mu_m \cdot \mathbf{B}_m) \Rightarrow f_m = \mathcal{M}(\mu_m) \mathcal{G}(\mathbf{B}_{\nabla, m}) \end{aligned}$$

where $\mathcal{M}(\mu_m) : \mathbb{R}^3 \times \mathbb{R}^{3 \times 5}$ represents a map of the field independent spatial gradients to forces on the dipole μ_m (Petruska and Nelson, 2015). The applied magnetic forces and torques together with the initial and a few time-evolved configurations are shown in **Figure 10**.

For modeling, we consider three nodes located at the locations of the proximal and distal magnets, and the clamped end of the rod. It should be pointed out that the performance of the proposed RNN-based model, unlike conventional algorithms, is independent of the number of nodes considered for the whole manipulator. Therefore, it is sufficient to model points

of interest. The idea is to independently manipulate each magnet (actuation point). However, the setup provides us with 8 degrees of freedom, meaning that positions and orientations (12 degrees of freedom) cannot be manipulated at the same time. Therefore, we carried out the experiment to achieve only orientation control.

669 1-by-3 position samples and 1-by-6 augmented wrench samples (i.e., $[\tau, f]^T$) for the both magnets are obtained. By choosing the size of history horizon 2 ($\eta = 1$), the augmented 2-by-6 position tensors are reshaped for each time step and fed to the model as the Input Layer I. The same preparation process is applied for the force data samples which are used as the Input Layer II. The prepared data-set is called Data-set IV and 60% of the data is used for training process. We suggest the same model for both moving nodes and the architecture of the model is



depicted in **Figure 11** which is the same for the proximal and distal nodes. The architecture shows the Input layers I and II consist of tensors of size $(Batch\ Size \times 2 \times 6)$. The first dimension of all layers are reserved for batch sizes and for the training, the batch size 1 was chosen. In the architecture of the model in **Figure 11**, the Input, Hidden and Output Layers I and II together with the number of nodes and type of layers are demonstrated according to **Figure 2B**.

5.3 RESULTS

The distal and proximal node rotations are predicted both by Cosserat rod model and the proposed model, and the results are shown in **Figure 12**. Also, the maximum and mean absolute errors are stated in an ordered pair in **Table 1**.

The computation time required to find a solution of the manipulator statics from a Boundary Value Problem (BVP) with Cosserat rod theory depends on the quality of the initial solution guess, i.e., $n(s)$ and $m(s)$ at $s = 0$, the tolerable error ($E \in \mathbb{R}$), and the number of nodes ($N \in \mathbb{N}$) used to discretize the manipulator.

A tolerable error describes the error between the distal internal forces and moments obtained from forward integration which are called n_f^d and m_f^d , and distal boundary condition, i.e., n_b^d and m_b^d . The tolerable error can be written as $[n_f^d - n_b^d, m_f^d - m_b^d]_2 \leq E$.

Decreasing the tolerable error increases the solution accuracy, but potentially requires more time to solve convex optimizations for the BVP. Increasing the number of nodes is necessary to describe complex manipulator geometries, but should be chosen to minimize the required steps during forward integration.

To visualize how the required computation time changes with the number of nodes and the tolerable error, multiple simulations were performed by assigning known torques τ_m and forces f_m for $m = 1, 2$, to the manipulator, and finding a valid solution from solving the BVP. Changes to tolerable errors (E) and number of nodes (N) were made manually. For example, an error of 2% in the initial solution guess was obtained by multiplying the valid solution with 0.98. After each change the BVP was solved again fifty times. The obtained mean and standard deviation of the computation times are shown in **Figure 13**. By taking into account all the aforementioned variables, i.e., number of nodes and the tolerable error, the Cosserat rod model is capable of

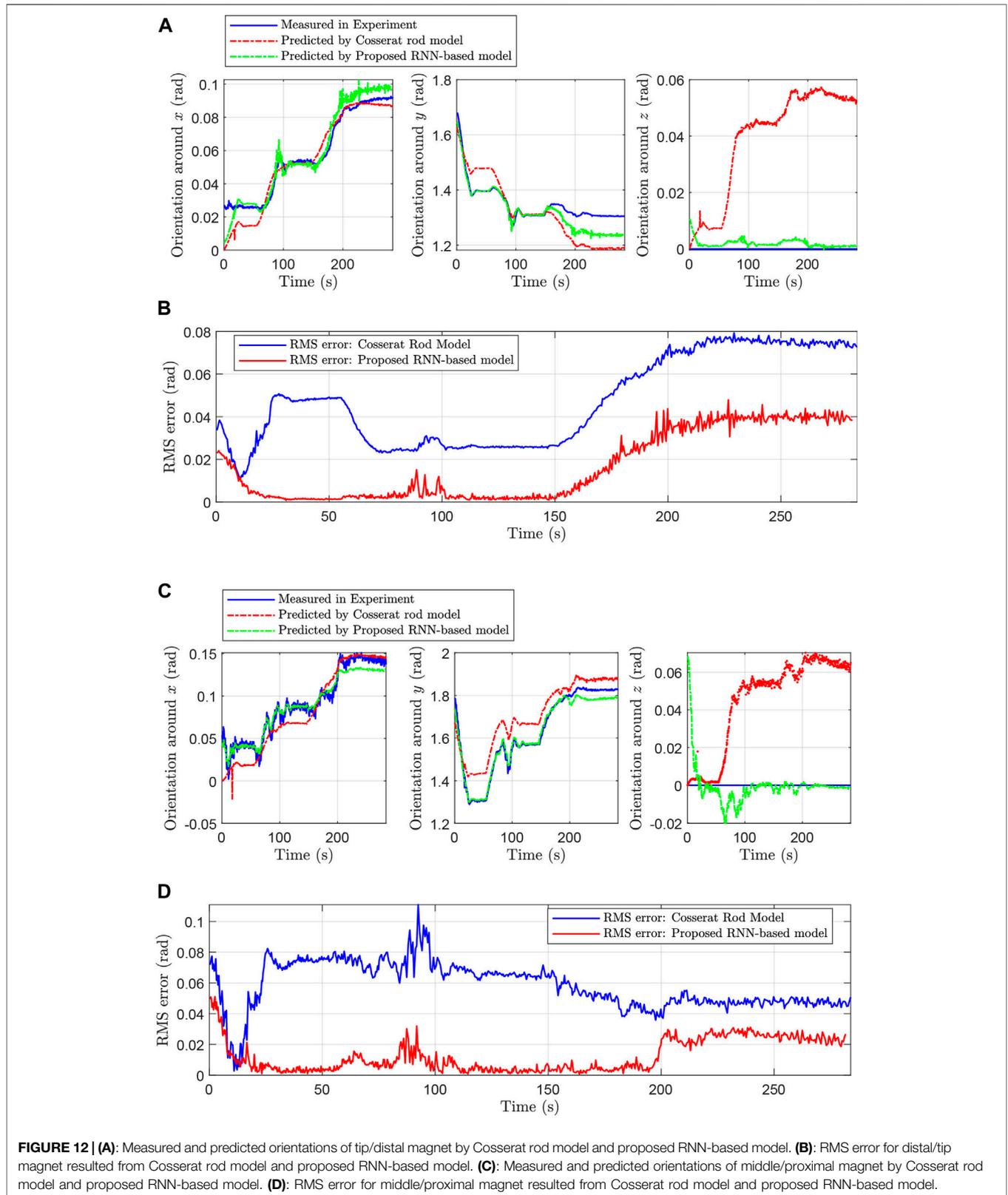


TABLE 1 | The maximum and mean absolute errors around the x, y, and z-axes in ordered pairs for the distal and proximal nodes.

	x-axis	y-axis	z-axis
Results for distal node			
Proposed RNN-based model	(1.27°, 0.23°)	(4.69°, 1.27°)	(0.60°, 0.10°)
Cosserat rod model	(3.56°, 0.82°)	(10.55°, 5.03°)	(4.02°, 2.22°)
Results for proximal node			
Proposed RNN-based model	(1.13°, 0.31°)	(3.30°, 0.89°)	(3.93°, 0.31°)
Cosserat rod model	(1.55°, 0.36°)	(7.22°, 3.51°)	(3.29°, 1.99°)

achieving real-time performances of the bandwidths between 8.33 and 50 Hz.

Figure 13A shows how the computation time required for solving a solution to the BVP changes with decreasing tolerable error (E) and increasing percentage errors from a valid solution (at 0%), for a constant number of nodes. Also, **Figure 13B** shows how the computation time required for solving a solution to the BVP changes with an increasing number of nodes (N) and increasing percentage errors from a valid solution, for a constant tolerable error. However, It should be mentioned that the proposed RNN-based model shows a real-time performance with a bandwidth of 60.75 Hz on average for the given architecture in **Figure 7**, number of epochs = 25, and batch size = 1. In addition, **Figure 14A** demonstrates the computation bandwidth required for the prediction of the next step using the trained model with a different number of LSTM units and a different size of time history horizons in Data-set IV. The figure suggests that computation bandwidths are fairly unchanged with the number of LSTM units; however, increasing the length of time history reduces bandwidth. The optimal region maximizing the bandwidth is approximately with time history size in (2, 20) and LSTM unit size in (5, 15). **Figure 14B** suggests that RMS error of the prediction decreases by increasing the number of LSTM units and the optimal area minimizing RMS errors is approximately with time history size in (2, 20) and LSTM unit size in (20, 25).

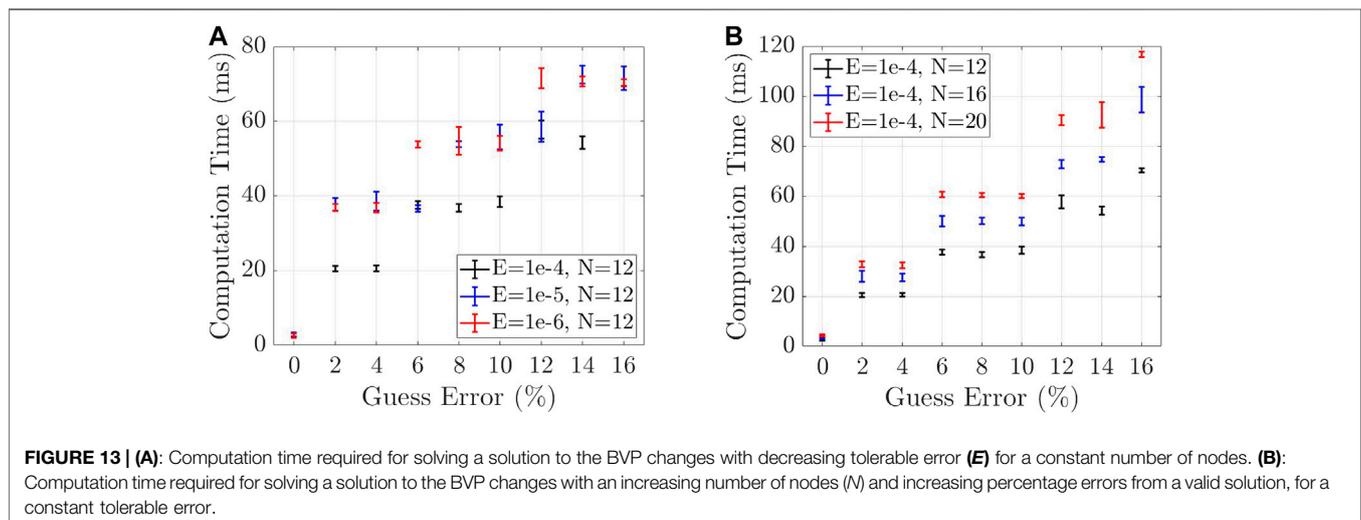
To sum up, this experiment demonstrates that not only can the presented RNN-based model outperform classical modeling approaches such as the Cosserat rod model, but also it shows possibilities to use the model in practice for closed-loop control applications.

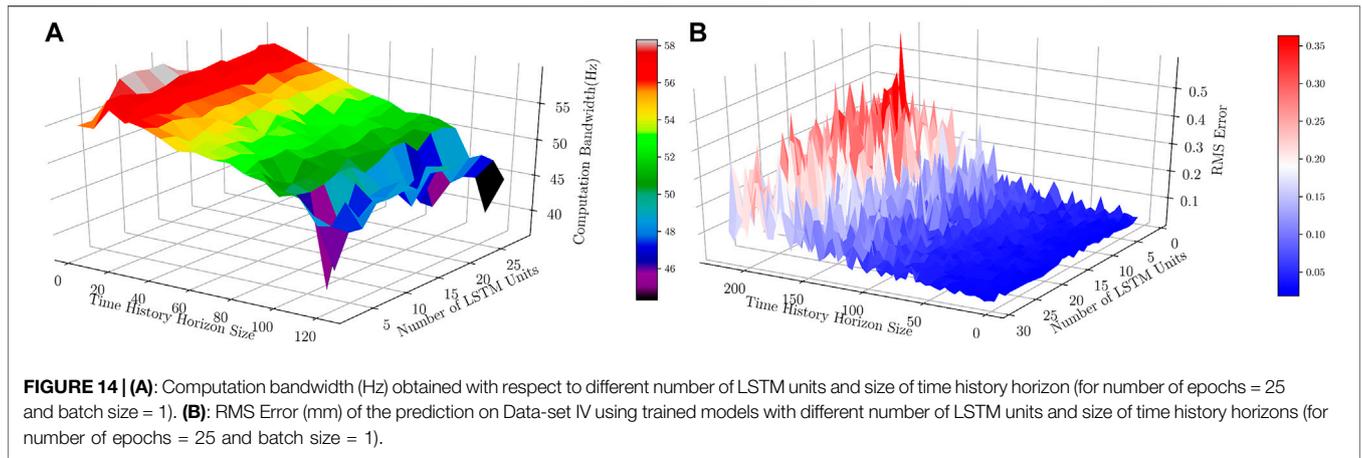
6 DISCUSSION

This work suggests a distributed architecture for modeling complex dynamical systems by using multiple light-weight RNN-based models. As a result, the architecture would be easier to design and debug, and also benefits from faster convergence compared to one large network. Furthermore, large networks may take longer times to be trained, and they may not show an acceptable performance and readjusting (hyper-)parameters and restarting the training process might be necessary.

Increasing the size of history horizons in training stages may reduce the error to some extent, but on the other hand, it makes the model slower. Based on conventional dynamical models, the length of the history size should be at least 2. To reach a state-of-the-art performance, i.e., having less error and faster model simultaneously, one may prefer varied batch sizes in the training and run-time phases. As a suggestion, we can use different batch sizes for training and run-time stages. A model can be trained with appropriate batch sizes such that the model performance suits the given criteria. Afterward, one can create a new network with the pre-trained weights compiled with a batch size of 1.

The performance, i.e., the convergence and stability, of the presented algorithm in this paper, unlike conventional algorithms, is independent of the number of nodes considered for the whole manipulator. To be specific, in the analytical model, there might be a need for several discretization nodes to achieve a convergent solution with a specific tolerable error; however, in the RNN-based model, only specific points/points of interests (e.g., two actuation points in the experiment) are considered. In other words, in the experiment, 13 nodes (4 for each flexible subsection





and two for each magnet, and 1 for the base) were chosen for solving the Cosserat rod model, but two nodes were selected for the RNN-based model. However, the complexity of dynamical systems (i.e., PDEs) affects the complexity of the architecture used in the RNN-based model, i.e., the number of layers and LSTM units and generally how deep the model is. Nevertheless, the suggested model suits parallel implementation and can benefit from a high bandwidth for closed-loop control applications. Furthermore, the architectures of the proposed RNN-based model can be optimized by reducing the number of layers and trainable parameters to maximize the achievable bandwidths.

The evaluations showed that incorporating poses of adjacent nodes and also wrenches as a separated input might help to have, to some extent, a generalizable model rather than just purely learning the structure of data. However, supervised learning methods likely tend to preserve structure of data, and these models might not entirely respect underlying physics (conservation laws). In other words, these methods might not be wholly physics-aware and applicable for untrained/unprecedented dynamics or geometries without any adjustment, re-training, or using techniques such as transfer learning, etc. One possible and interesting solution (Psychogios and Ungar, 1992; Lagaris et al., 1998; Raissi et al., 2019) to overcome this problem and move toward fully physics-aware neural networks is revisiting lost functions for the training process. To be specific, it is mentioned in Problem Statement Section that the idea is finding solutions for PDEs given in Eq. 1, i.e., $\Lambda(x, t)$ and $\phi(x, t)$ for sufficiently large number (e.g., N_f) of pair $(x_i, t_i) \in (0, L) \times [0, T]$ in which L is the unreformed length of the manipulator and parameter T is a user-defined time. Considering Eq. 1, a neural network can be learned by minimizing the mean squared error loss

$$\frac{1}{N_f} \left(\sum_{i=0}^{N_f} \|J\omega_{t_i} + \omega \times J\omega + n \times \Lambda^{-1}\phi_{x_i} - \Lambda^{-1}\Lambda_{x_i} \times m - m_{x_i} - \Lambda^{-1}\tau\|_{\{x_i, t_i\}}^2 + \sum_{i=0}^{N_f} \|M\phi_{t_i} - \Lambda(\Lambda^{-1}\Lambda_{x_i} \times n) - \Lambda n_{x_i} + f^{nc} - f\|_{\{x_i, t_i\}}^2 \right)$$

This modified loss function enforces the structure imposed by Eq. 1 for large number (e.g., N_f) of pair $(x_i, t_i) \in (0, L) \times [0, T]$ and the trained neural network will be aware of governing PDEs.

7 CONCLUSION

This paper describes an approach for the real-time prediction of dynamics for general continuum soft manipulators, based on machine learning techniques and Lie group variational integration methods. Poses of a soft, polymer-based manipulator, in the presence of conservative and non-conservative wrenches, are predicted and validated experimentally. The comparison results of the proposed model and a well-known model for continuum manipulators, i.e., Cosserat rod theory, are also provided, revealing the practical effectiveness of the proposed model. The presented method can be extended to different soft robots with different shapes and materials. In addition, training of physics-aware neural networks for solving PDEs and the procedure of a model-based controller design are topics of research to be studied as future work.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

AT: Formal analysis, Visualization, Writing original draft, VV: Review and Editing, MR: Writing, Analysis, OE: Review and Editing, JT: Review and Editing, KM: Review and Editing, SM: Review, Editing, and Resources for experiment, ØM: Review and Editing.

FUNDING

This research has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 Research and Innovation programme (Grant Agreement #866494 - project MAESTRO).

REFERENCES

- Ansari, Y., Manti, M., Falotico, E., Mollard, Y., Cianchetti, M., and Laschi, C. (2017). Towards the development of a soft manipulator as an assistive robot for personal care of elderly people. *Int. J. Adv. Robot. Syst.* 14, 172. doi:10.1177/1729881416687132
- Antman, S. S. (1995). "Nonlinear problems of elasticity," in *Applied mathematical sciences*. Editors J. E. Marsden and L. Sirovich (New York, United States: Springer), Vol. 107.
- Burgner, J., Swaney, P. J., Lathrop, R. A., Weaver, K. D., and Webster, R. J. (2013). Debulking from within: a robotic steerable cannula for intracerebral hemorrhage evacuation. *IEEE Trans. Biomed. Eng.* 60, 2567–2575. doi:10.1109/TBME.2013.2260860
- Demoures, F., Gay-Balmaz, F., Leyendecker, S., Ober-Blöbaum, S., Ratiu, T. S., and Weinand, Y. (2015). Discrete variational Lie group formulation of geometrically exact beam dynamics. *Numer. Math.* 130, 73–123. doi:10.1007/s00211-014-0659-4
- Edelmann, J., Petruska, A. J., and Nelson, B. J. (2017). Magnetic control of continuum devices. *Int. J. Robot. Res.* 36, 68–85. doi:10.1177/0278364916683443
- Grady, M. S., Howard, M. A., Dacey, R. G., Blume, W., Lawson, M., Werp, P., et al. (2000). Experimental study of the magnetic stereotaxis system for catheter manipulation within the brain. *J. Neurosurg.* 93, 282–288. doi:10.3171/jns.2000.93.2.0282
- Grazioso, S., Di Gironimo, G., and Siciliano, B. (2019). A geometrically exact model for soft continuum robots: the finite element deformation space formulation. *Soft Robot.* 6, 790–811. doi:10.1089/soro.2018.0047
- Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.* 9, 1735–1780. doi:10.1162/neco.1997.9.8.1735
- Hyatt, P., Wingate, D., and Killpack, M. D. (2019). Model-based control of soft actuators using learned non-linear discrete-time models. *Front. Robot. AI* 6. doi:10.3389/frobt.2019.00022
- Jones, B. A., Gray, R. L., and Turlapati, K. (2009). "Three dimensional statics for continuum robotics," in IEEE/RSJ International Conference on Intelligent Robots and Systems, St Louis, United States, October 11–15 2009. (New York, United States: IEEE), 2659–2664. doi:10.1109/IROS.2009.5354199
- Jung, J., Penning, R. S., and Zinn, M. R. (2014). A modeling approach for robotic catheters: effects of nonlinear internal device friction. *Adv. Robot.* 28, 557–572. doi:10.1080/01691864.2013.879371
- Katzschmann, R. K., Marchese, A. D., and Rus, D. (2015). Autonomous object manipulation using a soft planar grasping manipulator. *Soft Robot.* 2, 155–164. doi:10.1089/soro.2015.0013
- Kesner, S. B., and Howe, R. D. (2011). Position control of motion compensation cardiac catheters. *IEEE Trans. Robot.* 27, 1045–1055. doi:10.1109/TRO.2011.2160467
- Lagaris, I. E., Likas, A., and Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* 9, 987–1000. doi:10.1109/72.712178
- Lee, H., and Kang, I. S. (1990). Neural algorithm for solving differential equations. *J. Comput. Phys.* 91, 110–131. doi:10.1016/0021-9991(90)90007-N
- Mahvash, M., and Dupont, P. E. (2011). Stiffness control of surgical continuum manipulators. *IEEE Trans. Robot.* 27, 334–345. doi:10.1109/TRO.2011.2105410
- Marban, A., Srinivasan, V., Samek, W., Fernández, J., and Casals, A. (2019). A recurrent convolutional neural network approach for sensorless force estimation in robotic surgery. *Biomed. Signal Process. Control.* 50, 134–150. doi:10.1016/j.bspc.2019.01.011
- McMahan, W., Chitrakaran, V., Csencsits, M., Dawson, D., Walker, I. D., Jones, B. A., et al. (2006). "Field trials and testing of the OctArm continuum manipulator," in Proceedings 2006 IEEE International Conference on Robotics and Automation, Orlando, United States, May 15–19 2006 (New York, United States: IEEE), 2336–2341. doi:10.1109/ROBOT.2006.1642051
- Pai, D. K. (2002). Strands: interactive simulation of thin solids using cosserat models. *Comput. Graph. Forum* 21, 347–352. doi:10.1111/1467-8659.00594
- Petruska, A. J., and Nelson, B. J. (2015). Minimum bounds on the number of electromagnets required for remote magnetic manipulation. *IEEE Trans. Robot.* 31, 714–722. doi:10.1109/TRO.2015.2424051
- Psichogios, D. C., and Ungar, L. H. (1992). A hybrid neural network—first principles approach to process modeling. *AIChE J.* 38, 1499–1511. doi:10.1002/aic.690381003
- Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* 378, 686–707. doi:10.1016/j.jcp.2018.10.045
- Rucker, D. C., and Webster III, R. J. (2011). Statics and dynamics of continuum robots with general tendon routing and external loading. *IEEE Trans. Robot.* 27, 1033–1044. doi:10.1109/TRO.2011.2160469
- Sadati, S., Shiva, A., Elnaz Naghibi, S., Rucker, C., Renson, L., Bergeles, C., et al. (2019). "Reduced order vs. Discretized lumped system models with absolute and relative states for continuum manipulators," in Robotics: Science and Systems XV (Robotics: Science and Systems Foundation), Freiburg im Breisgau, Germany, June 22–26 2019. doi:10.15607/RSS.2019.XV.076
- Sikorski, J., Dawson, L., Denasi, A., Hekman, E. E. G., and Misra, S. (2017). "Introducing BigMag—a novel system for 3D magnetic actuation of flexible surgical manipulators," in IEEE International Conference on Robotics and Automation (ICRA), Marina Bay Sands Singapore, May 29, 2017 (Singapore: IEEE), 3594–3599. doi:10.1109/ICRA.2017.7989413
- Sikorski, J., Denasi, A., Bucchi, G., Scheggi, S., and Misra, S. (2019). Vision-based 3-D control of magnetically actuated catheter using BigMag—an array of mobile electromagnets coils. *IEEE/ASME Trans. Mechatron.* 24, 505–516. doi:10.1109/tmech.2019.2893166
- Simaan, N., Taylor, R., and Flint, P. (2004). "A dexterous system for laryngeal surgery," in IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04, New Orleans, LA, USA, April 26–May 1, 2004. 351–357. doi:10.1109/ROBOT.2004.1307175
- Tariverdi, A., Venkiteswaran, V. K., Martinsen, Ø. G., Elle, O. J., Tørresen, J., and Misra, S. (2020). Dynamic modeling of soft continuum manipulators using lie group variational integration. *Plos One* 15, e0236121. doi:10.1371/journal.pone.0236121
- Thuruthel, T. G., Falotico, E., Renda, F., and Laschi, C. (2017). Learning dynamic models for open loop predictive control of soft robotic manipulators. *Bioinspir. Biomim.* 12, 066003. doi:10.1088/1748-3190/aa839f
- Thuruthel, T. G., Shih, B., Laschi, C., and Tolley, M. T. (2019). Soft robot perception using embedded soft sensors and recurrent neural networks. *Sci. Robot.* 4, eaav1488. doi:10.1126/scirobotics.aav1488
- Till, J., Aloï, V., and Rucker, C. (2019). Real-time dynamics of soft and continuum robots based on Cosserat rod models. *Int. J. Robot. Res.* 38, 723–746. doi:10.1177/0278364919842269
- Turan, M., Almalioglu, Y., Araujo, H., Konukoglu, E., and Sitti, M. (2018). Deep EndoVO: a recurrent convolutional neural network (RCNN) based visual odometry approach for endoscopic capsule robots. *Neurocomputing* 275, 1861–1870. doi:10.1016/j.neucom.2017.10.014
- Weinan, E., Han, J., and Jentzen, A. (2017). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Commun. Math. Stat.* 5, 349. doi:10.1007/s40304-017-0117-6
- Wen, T., Wan, T. R., Gould, D. A., and John, N. W. (2012). Thien HowA stable and real-time nonlinear elastic approach to simulating guidewire and catheter insertions based on cosserat rod. *IEEE Trans. Biomed. Eng.* 59, 2211–2218. doi:10.1109/TBME.2012.2199319
- Xu, K., and Simaan, N. (2010). Intrinsic wrench estimation and its performance index for multisegment continuum robots. *IEEE Trans. Robot.* 26, 555–561. doi:10.1109/TRO.2010.2046924

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2021 Tariverdi, Venkiteswaran, Richter, Elle, Tørresen, Mathiassen, Misra and Martinsen. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

APPENDIX: COSSERAT ROD THEORY

The Cosserat rod model of the manipulator assigns a position $p(s) \in \mathbb{R}^3$, orientation quaternion $q(s) = (q_r, q_i) \in \mathbb{R}^4$, internal force $n(s) \in \mathbb{R}^3$, and internal moment $m(s) \in \mathbb{R}^3$ to a material cross-section at centerline position $s \in [0, L]$, where $L \in \mathbb{R}^+$ is the length of the manipulator, giving a material state vector $y = [p^T, q^T, n^T, m^T]^T$. A set of thirteen ordinary differential equations describe how the state vector evolves between centerline positions (Edelmann et al., 2017)

$$\begin{aligned} p' &= R(q)v, \\ q' &= \frac{1}{2} \begin{bmatrix} -q_i^T \\ q_r I_3 - [q_i]_\times \end{bmatrix} R(q)u, \\ n' &= -f, \\ m' &= -p' \times n - \tau, \\ v &= K_s^{-1} R(q)^T n + \hat{v} \end{aligned}, \quad u = K_b^{-1} R(q)^T m + \hat{u},$$

where $p' \equiv \partial_s p$, $[\cdot]_\times : \mathbb{R}^3 \rightarrow \mathbb{R}^{3 \times 3}$ a mapping to a skew-symmetric matrix, $R(q) \in SO(3)$ the rotation matrix associated with orientation quaternion q , $K_s, K_b \in \mathbb{R}^{3 \times 3}$ diagonal shear and bending stiffness matrices, $I_3 \in \mathbb{R}^{3 \times 3}$ a unit matrix, $v(s) \in \mathbb{R}^3$ and $u(s) \in \mathbb{R}^3$ the material strain and bending, and $\hat{v}(s) = [0, 0, 1]^T$ and $\hat{u}(s) = [0, 0, 0]^T$ the intrinsic

material strain and curvature. External forces $f(s) \in \mathbb{R}^3$ and torques $\tau(s) \in \mathbb{R}^3$ determine the shape of the manipulator (Antman, 1995; Rucker and Webster, 2011). The manipulator is subject to a distributed gravity force (not shown in **Figure 9**) and magnetically exerted distributed forces f_m and torques τ_m due to interaction of the magnets with the magnetic field B_m , where $m = 1, 2$ denotes the magnet index from the base of manipulator. Given the exerted torques and forces, the shape of the manipulator is solved as a Boundary Value Problem (BVP). The base of the manipulator is fixed to a rigid base with constant position p_0 and orientation q_0 , and its distal tip is free with constant internal force n_L^d and moment m_L^d . The proximal and distal boundary conditions are then formulated as follows, assuming no tip wrench, we have $\mathbb{P}(y_0) = [p_0^T, q_0^T]^T$ and $\mathcal{D}(y_L) = [n_L^{dT}, m_L^{dT}]^T = 0$. The BVP is solved with a forward integration using an explicit Runge–Kutta fourth order method, and convex optimization using Levenberg–Marquardt (Till et al., 2019). The unknown proximal state parameters $\xi = [n_0^T, m_0^T]^T$ are guessed and subject to the optimization where $N \in \mathbb{N}$ is the number of discrete steps along s . Then y_N are the manipulator distal state parameters obtained from the forward integration using an explicit Runge–Kutta fourth order method. The error between desired distal boundary condition $\mathcal{D}(y_L)$ and $\mathcal{D}(y_N)$ determines if the solution (ξ) is accepted.