Received 5 August 2022, accepted 30 August 2022, date of publication 5 September 2022, date of current version 14 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3204652

RESEARCH ARTICLE

Modeling Fabric-Type Actuator Using Point Clouds by Deep Learning

YANHONG PENG, (Member, IEEE), HIROKI YAMAGUCHI, YUKI FUNABORA, (Member, IEEE), AND SHINJI DOKI, (Senior Member, IEEE)

Department of Information and Communication Engineering, Graduate School of Engineering, Nagoya University, Nagoya, Aichi 464-8603, Japan

Corresponding authors: Yanhong Peng (yhpeng@nagoya-u.jp) and Yuki Funabora (funabora@nagoya-u.jp)

This work was supported by the Japan Science and Technology Agency (JST), PRESTO, Japan, Grant Number JPMJPR185A and JST Fusion Oriented REsearch for disruptive Science and Technology (FOREST) Program, Japan, Grant Number JPMJFR216T.

ABSTRACT Flexible actuators are popular in the consumer and medical fields because of their flexibility and compliance. However, they are typically difficult to model because of their viscoelasticity and nonlinearity. This letter proposes a method for correcting the deformation of the simulated flexible robots to make it similar to the deformation of real robots using point clouds by deep learning. Long short-term memory (LSTM) can simulate the next frame of actuator deformation from the previous frames of deformations. In this study, we presented the robots with four different muscle structures. We found that using an encoder–LSTM–decoder network can improve the similarity between the deformation of a learned muscle structure and the real deformation and is also effective in correcting the deformation of the unlearned structures. Our correction method reduced the average Chamfer distance of the simulated point clouds of the basic-type structure actuator from 15.89 to 7.81. This research can provide a new concept for future flexible robot modeling using point clouds.

INDEX TERMS Soft actuators, deep learning, modeling for soft robots, McKibben Artificial muscle, point cloud.

I. INTRODUCTION

Soft actuators, mainly made of stretchable and flexible materials, have received considerable attention in recent years because of their superior human–machine interface to hard robotics, which has substantial economic benefits [1], [2], [3], [4]. They have many uses in wearable devices [5], [6], [7], [8] and have also been investigated for potential applications in medical rehabilitation equipment [9], [10], [11], [12]. However, there are common limitations in modeling and controlling soft actuators because the material's structural compliance, nonlinearity action, and the viscoelasticity in the material result in complex and unpredictable behaviors [13]. Over the past several decades, deep learning has made unprecedented progress. The insights that artificial intelligence technology can extract from complex data benefit the field of medicine [14], [15], [16], autonomous driving [17], [18], [19], [20], and many other fields. Deep learning is also well-known for effectively solving nonlinear problems in soft robotics [21]. The finite element method (FEM) and position-based dynamics (PBD), as current mainstream methods, have some limitations in modeling, such as the balance between computing power consumption and model accuracy; nonetheless, and deep learning can compensate for these shortcomings to some extent [22], [23], [24], [25], [26], [27].

This paper presents a method of modeling four soft fabric-type actuators by correcting the simulated point clouds through deep learning to make them approach the point clouds obtained from real actuators using four depth sensors. This paper also aims to solve the above limitations of fabric-type actuators by deep learning. PointNet [28] is a robust framework that can transform the three-dimensional (3D) coordinates of point clouds into local or global features. It has achieved excellent performance recently in point cloud classification [29], [30], [31] and semantic segmentation [32], [33], [34], [35] tasks. In this research, the encoder of PointNet was used to extract the global features of the point clouds, and then the simulated point cloud features of the previous

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

five relevant frames were learned by long short-term memory (LSTM) as time-series data to predict the real point cloud features of the next frame. Finally, using the decoder provided by Charles [36], we can decode the real point cloud features into 3D point cloud coordinates. By establishing a neural network structure of encoder–LSTM–decoder, we can describe the real point cloud of an actuator from the simulated point cloud of the previous state. In this process, the neural network compensates for nonlinear mechanical actions from the soft actuator design material and geometry difficult to simulate with point clouds. The similarity between the corrected simulated point cloud and the real point cloud was significantly improved.

II. RELATED WORK

FEM, PBD, and deep learning are typically used as simulation methods for flexible actuators. FEM has high simulation accuracy and is used not only used for structural analysis and material property simulation but also for the deformation simulation of flexible objects [37], [38]. However, the computational cost increases when simulating flexible objects accurately, and the real-time simulation is challenging. For example, Dang *et al.* [39] simulated the radial shrinkage of flexible actuators with slight deformation and no timing information by FEM, achieving a considerable simulation accuracy but with high computing power requirements.

PBD is a method commonly used in computer graphics and 3D games with low computational costs and is suitable for real-time simulation [40], but the simulation is not as accurate as the FEM, and there are some differences between simulated actuators and actual soft actuators. Moreover, the point cloud data can be processed well with PBD. Liu *et al.* [41] combined 3D visual perception with PBD modeling and proposed a real-to-simulated point cloud registration method. The PBD method was used to simulate soft tissue dynamics and rigid tool interactions for model-based control. Vision-based strategies were used to generate 3D reconstructed point cloud surfaces based on real-world operations to register and update simulations. This approach had been tissue-experimented on the da Vinci Research Kit, achieving higher accuracy than fusion-based reconstructions in occluded regions.

Liu *et al.* [42] developed a model based on PBD to achieve accurate trajectory torque estimation of rigid robots and the ability to support spring stiffness estimation. However, it is difficult to apply this method to soft robots. As the simulation of PBD is reasonable in motion but not always realistic in physical action [43], it is difficult to physically explain specific parameters on many occasions, especially for flexible bodies with nonlinear motion.

Deep learning can model the actuators combined with PBD to improve the accuracy. In our proposed method, the fabric-type actuator was based on Flexible Fabric Actuator [44] published by our laboratory and its variants, and the deformation of the fabric actuator was simulated iteratively including time-series data. Because it is necessary to reduce the computational cost of every deformation simulation, we used a PBD-based simulator. PBD-based simulators can simulate the simple fabric deformations caused by wind shaking or collisions with other objects, but the simulation accuracy is insufficient for complex nonlinear deformations [41]. The fabric-type actuator, which is the subject of this study, placed multiple artificial muscles on a fabric. As the artificial muscles contract, the fabric can deform to the expected shape. However, the contact between the fabric and the artificial muscles created a nonlinear mechanical effect when the actuator deformed. The simulation accuracy was degraded by nonlinear mechanical effects, and there was a large difference between the simulation results and the actual deformation of the actuator. Therefore, after using the PBD-based simulator to simulate the deformation of the actuator, we used a deep learning-based correction system to correct the nonlinear mechanical actions that the PBD-based simulator could not simulate. In deep learning, LSTM has many advantages over feedforward networks for nonlinear system modeling and is widely used in different engineering fields [45]. For example, Fan *et al.* [46] recently used LSTM to model nonlinear distributed thermal processes, achieving reliable simulated temperature distributions. In addition, Li *et al.* [47] developed a reduced-order model for a wind-bridge interaction system by LSTM to accurately and efficiently predict bridges' flutter and post-flutter behavior, with nonlinear unsteady aerodynamics induced by bridge motion. In this study, LSTM plays the primary role of correction because it can process time-series data and is suitable for connecting high-dimensional features (global features) of simulated and actual point clouds. With an appropriate correction method, the real deformation of the fabric-type actuator can be simulated.

The main contribution of this research is to combine deep learning with traditional simulation instead of solely using deep learning modeling. The correction system saved more computing power than the conventional modeling FEM and is more controllable than deep learning-based black-box models because the muscle distribution and parameters of the actuator were set in Unity. This research can provide rapid simulation close to real actuators for future robot design and provide an alternative solution for future real-time simulation.

III. ROBOT SETTING AND POINT CLOUD COLLECTION

A. BUILD THE ROBOT SYSTEM

There are three basic elements of the fabric-type actuator: the fabric body, the artificial muscle, and the fixed point. The fabric body promotes 3D deformation of the actuator by changing the force direction of the artificial muscle. The artificial muscle contracts in the axial direction of the muscle when air pressure is applied to it. The fixed point transmits the force of the artificial muscle to the fabric body by limiting their relative movement.

In this study, the square fabric body with a side length of 0.2 m was woven tightly by a thin flexible rubber

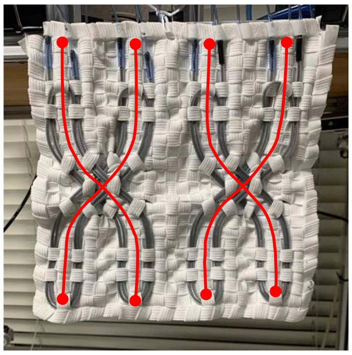


FIGURE 1. **Muscle configuration example: The red curve denotes the muscle pattern on the front side of a robot, and the muscles on the back of the robot are the same size as those in the front.**

swath (LYCRA® Soft Elastic Tape, 10.5 mm width, DAISO JAPAN). A thin McKibben artificial muscle (EM20, s-muscle Co.Ltd.) [48], [49] with a length of 800 mm and a tube outer diameter of 2.0 mm was passed through the flexible rubber swath interstice to make a fixed point, and air pressure of up to 400 kPa was applied. Fig. 1 depicts an example of muscle configuration. Moreover, various deformations and control patterns are shown in Fig. 2.

The fabric-type actuator was controlled by contracting and stretching a plurality of artificial muscles. Thus, a control system is required to independently control the contraction and extension states of each artificial muscle. The outline of our constructed control system is shown in Fig. 3. The control program was implemented in C ++ using Microsoft® Visual Studio Community 2017 and performed serial communication with an input–output device (USB3114, Measurement Computing Corporation). The input–output device received a command from the workstation and adjusted the voltage applied to an electro-pneumatic regulator (CRCB-0135W, CRCB-0136W, Koganei Corporation). The electro-pneumatic regulator adjusted the air pressure according to the applied voltage and air pressure to the artificial muscle. With this control system, the pressure applied to the artificial muscle can be controlled with a resolution of 0.1 kPa, and the control cycle was 50 ms.

B. BUILD THE SIMULATION SYSTEM

The fabric and the artificial muscle models were developed and combined by a fixed point to form a fabric-type actuator model. Unity® was used to simulate movements caused by the contraction of artificial muscles and the deformation of the actuator. To simulate the movement of the fixed point following the contraction of the artificial muscle model, the fabric and the artificial muscle models must interact.

Nvidia Flex [50], [51] was used to simulate a piece of fabric. It is particle-based physical simulation technology that can use particles to represent rigid bodies, fluids, and flexible objects. In Nvidia Flex, the fabric was represented as a group of constrained particles, which were placed at the vertices of the triangular polygons that make up the fabric model. The larger the number of particles, the easier the fabric can be

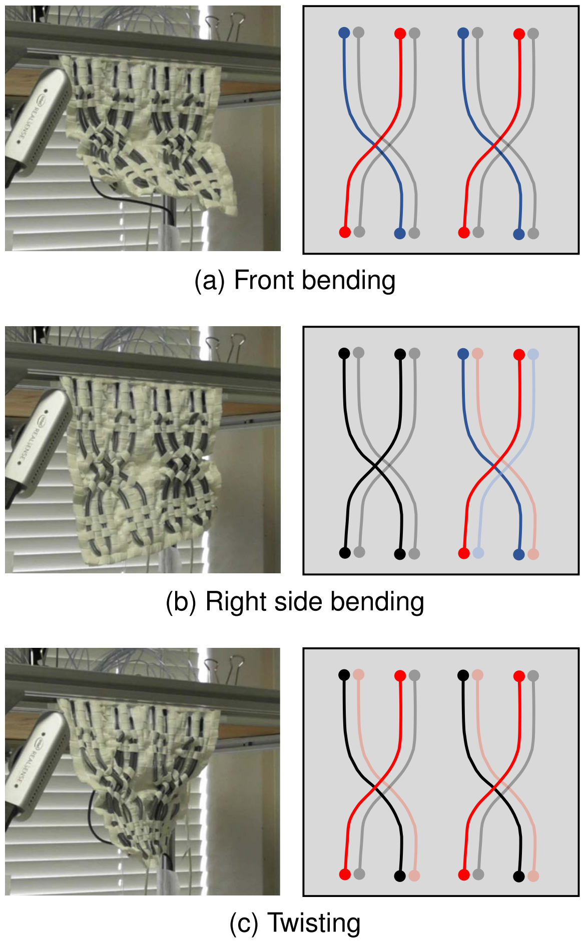


FIGURE 2. **Fabric-type actuator can achieve different motions through different muscle combinations. The black curves indicate the relaxed muscles (setting supplied air pressure to zero). The blue and red curves denote the activated muscles in different directions. The opaque colors represent the muscles in the front of the actuator, whereas the translucent colors represent the muscles in the back. (a) Front bending motion and its control pattern. (b) Right-side bending motion and its control pattern. (c) Twisting motion and its control pattern.**

described, but the higher the calculation cost. The size of the fabric model (Fig. 4(a)) was set to 1.00 m in length and width, and 0.01 m in thickness, considering the ease of processing objects on Unity.

In addition, the artificial muscle was created by connecting multiple tiny rigid body models to form a constrained chain. The size of each tiny rigid body model is a cuboid with a length of 0.01 m, a width of 0.01 m, and a height of 0.05 m. The tiny rigid body model was connected by a Configurable Joint option in Unity to realize the contraction of the artificial muscle. The local coordinate axes of the tiny rigid body model with Configurable Joint are depicted in Fig. 4(b). When no air pressure is applied to the artificial muscle, the movement of the rigid body model in the local y-axis direction is restricted to zero. When air pressure is applied, the movement of the rigid body model in the local y-axis direction can be restricted to a distance that matches the contraction rate of the actual artificial muscle, as described in [52] and [53]. The tiny rigid body model can be moved

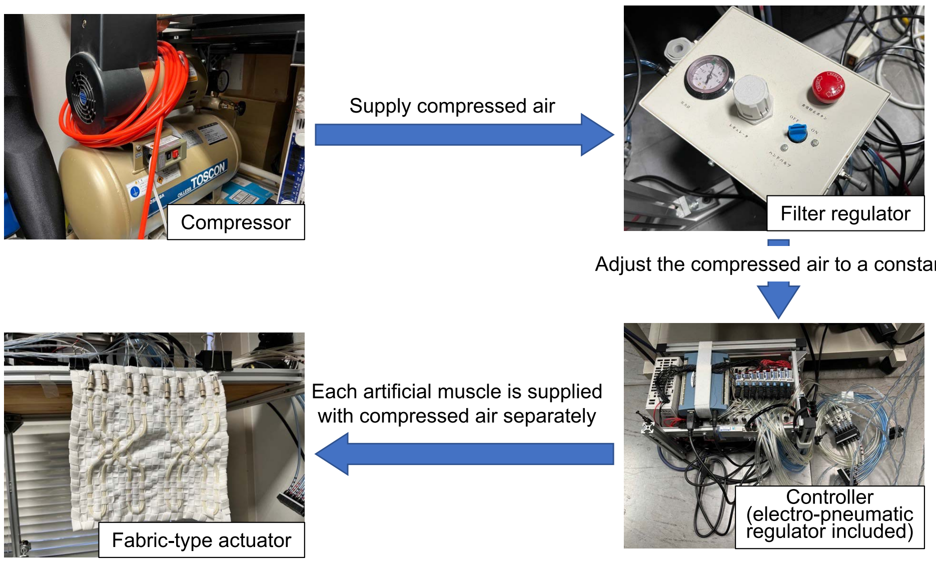

Supply compressed air
Compressor
Filter regulator
Adjust the compressed air to a constant
Each artificial muscle is supplied with compressed air separately
Fabric-type actuator
Controller (electro-pneumatic regulator included)

FIGURE 3. The robot control system comprises an air compressor that supplies air pressure to power artificial muscles and a regulation unit that includes a filter regulator to maintain air pressure. In addition, it comprises an electro-pneumatic regulator that controls air pressure supplied to individual artificial muscles, thereby controlling the deformation of the fabric actuator.

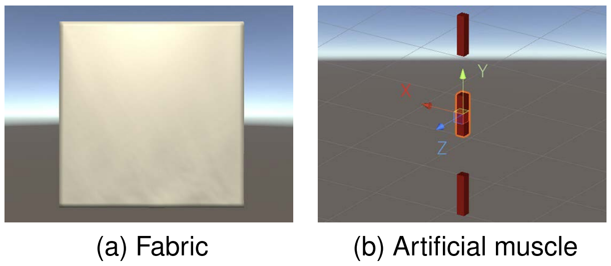


FIGURE 4. Simulated model representing the elements.

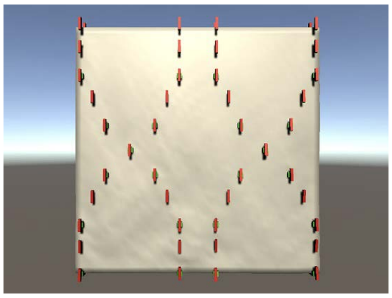


FIGURE 5. The simulated actuator model that combines the fabric and artificial muscle models.

in the positive direction of the local y-axis by applying a force. As a result, the artificial muscle model's tiny rigid body model moved to a restricted position and contracted as an artificial muscle model. After combining the fabric and artificial muscle models, the simulated actuator model (Fig. 5) was obtained.

C. POINT CLOUD ACQUISITION

A depth sensor (RealSense D435i, Intel Corporation) was used to measure the deformation of the actual robot. The measurement environment is constructed, as shown in Fig. 6(a), and the depth sensors are circled in red and located at the corner of the metal frame. As the occlusion may occur with one sensor, the number of point clouds collected from the actual actuator differed in each deformation, and we measured the entire fabric-type actuator without omission by setting four sensors from multiple angles. By converting the point cloud obtained from each sensor into the world coordinate system based on the arrangement of each sensor and the iterative closest point (ICP) algorithm [54], the deformation of the actuator can be acquired as a completed point cloud consisting of point clouds from different cameras. In the ICP algorithm, we can find the translation T and rotation R that minimize the sum of the squared error in Equation (2) from the two corresponding point sets P and X in Equation (1):

$$\begin{aligned} P &= \{p_1, p_2, \ldots, p_n\} \\ X &= \{x_1, x_2, \ldots, x_n\} \end{aligned} \tag{1}$$

$$E(R, T) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|x_i - Rp_i - T\|^2 \tag{2}$$

where x_i and p_i denote the corresponding points, and N_p represents the number of points. This computing process was completed using MATLAB® Computer Vision Toolbox. After the ICP registration, we downsampled the registered point clouds to approximately 3,800 points through a box grid filter. To facilitate the comparison of the difference between the actual and the simulated point clouds in the future, we randomly sampled the downsampled point clouds to make them be the same number as the simulated point clouds (3,362 points). The actuator and the four depth sensors were fixed on the metal frame to prevent changes in relative position.

The number of particles placed on one side of the simulated fabric-type actuator model was 41 × 41. Therefore, the total number of particles on the surface of the actuator model representing the point cloud was 41 × 41 × 2, totaling 3,362. The created fabric model with a point cloud is depicted in Fig. 6(b).

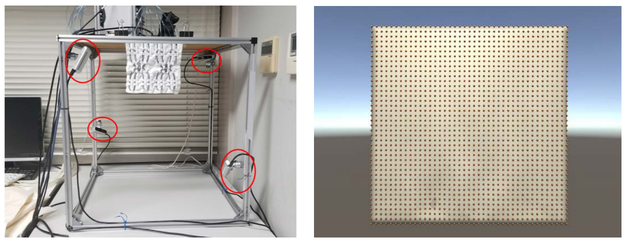


FIGURE 6. Point cloud acquisition. (a) After the shape point cloud of the robot is collected by the four depth sensors in all directions, the robot can be used to synthesize a complete point cloud using the ICP method. (b) Generate point clouds on both sides of the fabric to represent the fabric deformation directly in Unity.

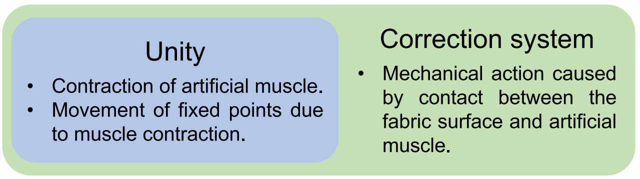
IV. CORRECTION SYSTEM DEVELOPMENT

A. CORRECTION STRATEGY

Some modeling errors are unavoidable in the simulation even though we attempted to adjust the parameters of the simulated actuator's properties based on the motion of the real actuator. The difference in deformation between the real robot and the simulated actuator is caused by the mechanical action that occurs between the fabric and the artificial muscle. However, the information is insufficient if the feature simulated point cloud and the actual point cloud with the same frame are directly connected with a neural network because the information about the moving process is missing. The time-series data can be used to describe the moving process information, such as the contraction process of the artificial muscle for some time and the physical effect of the muscle friction on the fabric. Therefore, a correction system that estimates the actual deformation of the actuator (the point cloud of the real actuator) from the deformation of the simulated actuator (the position of particles of the simulated actuator model) in time series can be developed to address the difference in deformation. In this study, Unity realized the contraction of artificial muscles and the movement of fixed points, and the correction system corrected the mechanical action that occurs between the fabric surface and the artificial muscles. This relation is shown in Fig. 7. The correction system was constructed by deep learning because the mechanical action that occurs between the fabric and the artificial muscle includes a temporal element, which can be corrected using time-series information. The correction system simulated the deformation of the $(k+1)$-th frame using the deformation of the l frame (from the $(k-l+1)$-th frame to the k-th frame).

B. THE STRUCTURE OF NEURAL NETWORK

Because the correction system predicted the frame using the past l frames, LSTM can be used to process time-series information. Unlike the images, the order of indexing of point clouds is irrelevant. In this study, a point cloud was converted to a feature dimension, which is not affected by the order of points, and features were associated with LSTM. The structure of the correction system comprised an **encoder**, a **decoder**, and the **LSTM**, as shown in the *Correction system Module* of Fig. 8.



Unity
• Contraction of artificial muscle.
• Movement of fixed points due to muscle contraction.
Correction system
• Mechanical action caused by contact between the fabric surface and artificial muscle.

FIGURE 7. Correction strategy.

According to PointNet [28], *Autoencoder*1 was trained by simulated point clouds (*Autoencoder*1, Fig. 8) and took its encoder as the encoder of the correction system. Furthermore, *Autoencoder*2 was trained by the real point clouds (*Autoencoder*2, Fig. 8) and took its decoder as the decoder of the correction system.

The encoder of *Autoencoder*1 converted point clouds with m points from the simulation into n dimensional features, and the LSTM associated the features of the simulated deformation of the continuous l frames with the features of the predicted deformation. The decoder converted the output with n-dimensional features from the LSTM into a point cloud with m points. This process is shown in the *LSTM Module* of Fig. 8.

Finally, the encoder (created by learning the simulated point cloud in *Autoencoder*1)–LSTM–decoder (created by learning the real point cloud n *Autoencoder*2) was connected to construct the correction system. This system can significantly correct modeling errors because of viscoelasticity and nonlinear movement.

C. LOSS FUNCTION FOR CORRECTION SYSTEM

Equation (3) was used to evaluate the loss function, and training was stopped when the loss function stopped improving on the validation set. Early stopping had two main parameters: min delta and patience. An absolute change in loss function less than min delta will be considered no improvement, and it was set to 0.1 in this study. Patience is the number of epochs with no improvement, after which training will be stopped. Patience was set to 10 when training *Autoencoder*1 and *Autoencoder*2, whereas it was set to 30 when training LSTM because the training curve of LSTM drops more slowly. After the correction system had been trained, we evaluated whether the corrected simulated point cloud can accurately describe the actual point cloud. Chamfer Distance (CD) [55] was used to evaluate the extent to which the simulated point cloud can fit the actual point cloud. In addition, the loss function of our neural network was also based on the CD, and the neural network was continuously trained toward the smaller value of the CD. In this study, the CD was defined using Equation (3) and was an average index showing the similarity between two groups of point clouds:

$$d_{CD}(S_1, S_2) = \frac{1}{|S_1|}\sum_{x\in S_1}\min_{y\in S_2}\|x-y\|_2^2 + \frac{1}{|S_2|}\sum_{y\in S_2}\min_{x\in S_1}\|x-y\|_2^2 \tag{3}$$

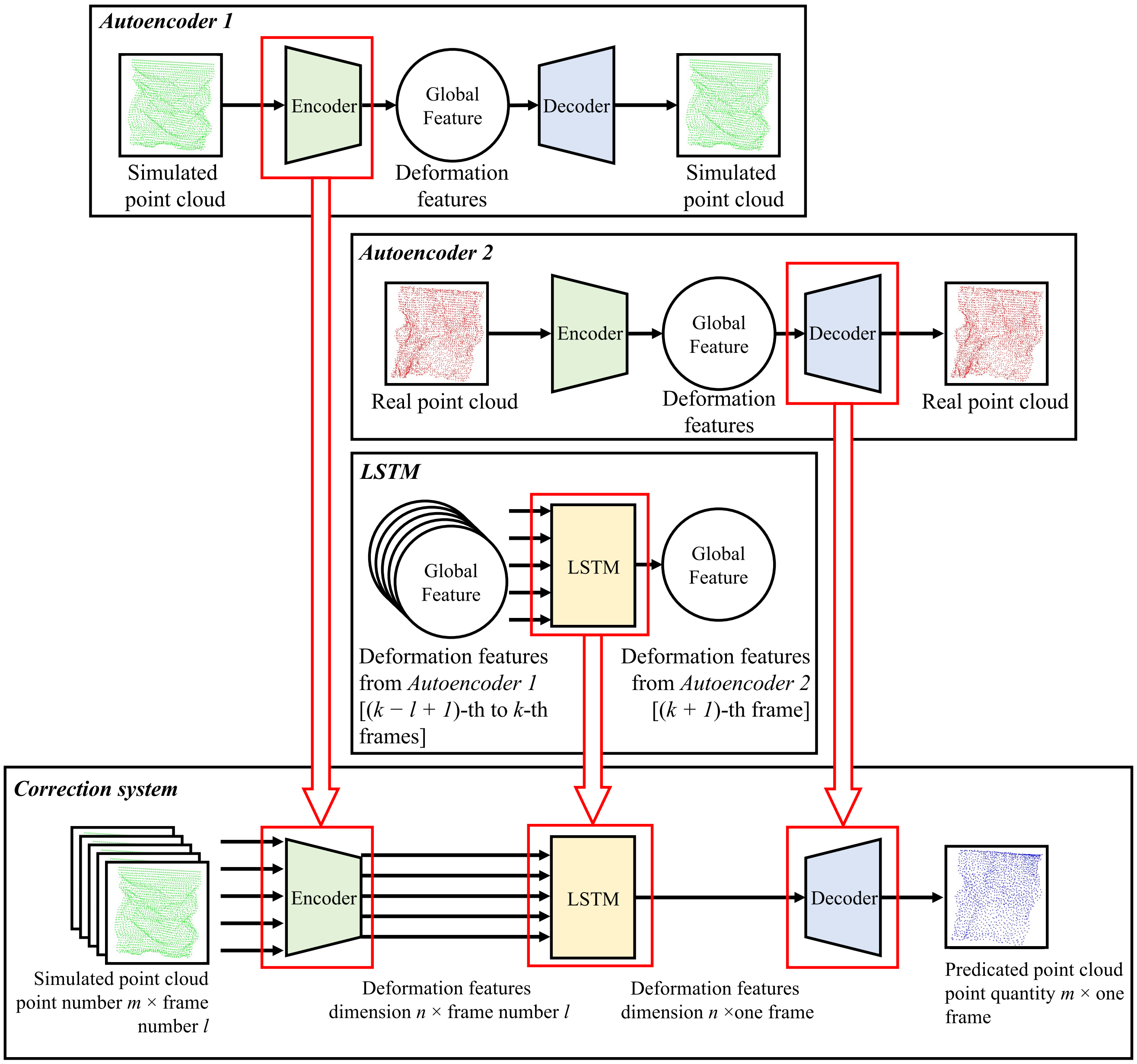

Autoencoder 1
Encoder
Global Feature
Decoder
Simulated point cloud
Deformation features
Simulated point cloud
Autoencoder 2
Real point cloud
Encoder
Global Feature
Deformation features
Decoder
Real point cloud
LSTM
Global Feature
LSTM
Global Feature
Deformation features from Autoencoder 1 [(k − l + 1)-th to k-th frames]
Deformation features from Autoencoder 2 [(k + 1)-th frame]
Correction system
Simulated point cloud point number m × frame number l
Encoder
Deformation features dimension n × frame number l
LSTM
Deformation features dimension n ×one frame
Decoder
Predicated point cloud point quantity m × one frame

FIGURE 8. The encoder (trained in *Autoencoder* 1) can encode the input simulated point cloud into a global feature, then LSTM converts the global feature of the previous frames into the global feature of the next frame, and finally the decoder (trained in *Autoencoder* 2) can decode the global feature into the corrected point cloud.

where S_1 and S_2 denote the two groups of point clouds, and x and y represent the coordinate values. The smaller value of the CD value, the more similar the two groups of point clouds are.

V. EXPERIMENT

A. ACTUATOR SETTING

The top of the fabric-type actuator was fixed with a cubic frame and deformed in a suspended state. The deformation of the actuator to be measured always started from a relaxed state. After the deformation was complete, the applied pressure was set to 0 kPa (the relaxed state). The air pressure applied to each artificial muscle was randomly determined from 0 kPa to 400 kPa with a step of 100 kPa. The measurement frame rate was set to 60 Hz, and a time-series point cloud for 90 frames was acquired from the start of the application of the pneumatic pattern. The four types of artificial muscle actuator configurations used for measurement are shown in Fig. 9, and Fig. 10 depicts an example of basic-type robot measurement results. The deformation of the actuator was described by point clouds measured using four depth sensors.

The side type was obtained by rotating the basic type 90° counterclockwise. There were subtle differences because the basic and side types have different camera angles, even with the same control input. Generally, tilting the actuator at a certain angle can be used as a type of learning data augmentation in deep learning because convolutional neural

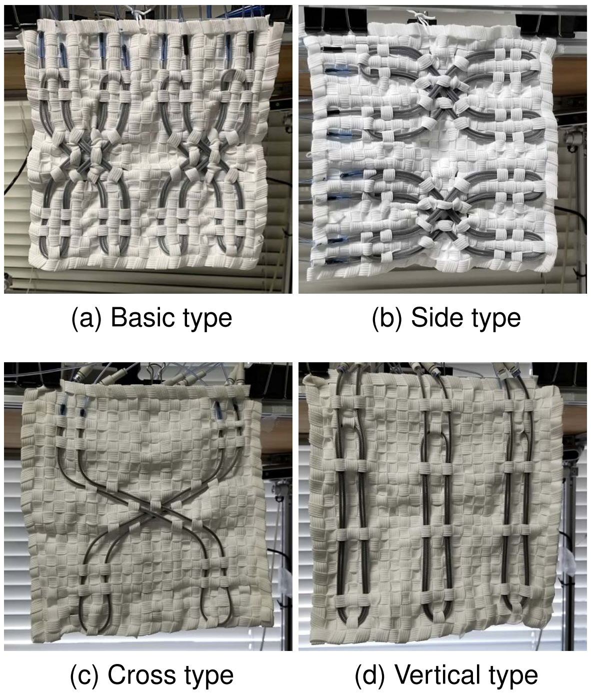


FIGURE 9. (a) Eight artificial muscles, four on the front side and four on the backside, are placed vertically. (b) Eight artificial muscles, four on the front side and four on the backside, are placed horizontally. (c) Two artificial muscles on one side are placed crosswise. (d) Three artificial muscles on one side are placed vertically.

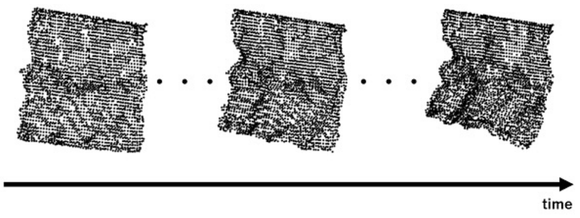


FIGURE 10. Actuator movements can be described as time-series point clouds.

networks (included by PointNet) do not contain rotation invariance [56]. The result of the side-type deformation can reveal whether the result can be corrected, even after the actuator is tilted. The cross-type deformation contained different muscle structure information obtained from the basic- and side-type deformation, which was used as learning data to improve the model's generalization ability. The vertical type was not involved in the neural network learning stage and was only used to test the performance of the correction system in the unknown muscle structure.

B. DATA PREPARATION

In this experiment, we collected 1,188 groups of basic-type deformations, 198 groups of side-type deformations, 25 groups of cross-type deformations, and 125 groups of vertical-type deformations. Each group of deformation contained 90 frames of point cloud data of motion (the entire motion was completed in 90 frames), which was collected at a sampling frequency of 60 Hz after the artificial muscle model started to run. The frame data allocation is shown in TABLE 1, each frame of point cloud data was randomly

TABLE 1. Data allocation.

	Train set	Validation set	Test set
Basic type	80,190	8,910	17,820
Side type	8,019	891	8,910
Cross type	2,025	225	-
Vertical type	-	-	11,250

shuffled. The number of deformations in the cross and vertical types was significantly smaller than that of the basic type because there were fewer variations in deformation with only one side installed artificial muscles. Moreover, the vertical-type actuator was used only for testing, whereas the cross-type actuator was used only for training. The number of deformations and frames obtained from the simulation was the same as the number of real deformations under each configuration. According to the number of particles in the fabric model in the simulation, the number of point clouds in each frame was processed into 3,362 points.

C. LEARNING BY NEURAL NETWORK

In this study, TensorFlow 2.7 was used to construct a the neural network. In the first step, we used the simulated point cloud as the training data and label data identically in *Autoencoder*1: the input and target data are the same. Then, the global feature of the simulated point cloud can be obtained by the encoder of *Autoencoder*1. In the second step, we used the real point cloud as the training and label data identically in *Autoencoder*2, and the global features of the real point cloud can be obtained by the encoder of *Autoencoder*2. The learning rate of 0.001 with the Adam optimizer was used in the training process of *Autoencoder*1 and *Autoencoder*2. In addition, a dropout layer with a dropout rate of 0.05 was used in the hidden layer in the decoder of *Autoencoder*1 and *Autoencoder*2 to improve the generalization ability of the simulated point cloud. In the third step, LSTM was employed to predict the global features of real point clouds from the global features of simulated point clouds. The training data were the global features of the simulated point cloud obtained by *Autoencoder*1, and the label data were the global features of the real point cloud obtained by *Autoencoder*2, as previously discussed. The learning rate of 0.00001 with the Adam optimizer was used in the training process of LSTM. LSTM was learned by using the global features converted by the above encoder and decoder. In this study, m = 3,362, n = 1,024, and l = 5. Finally, the encoder of *Autoencoder*1, LSTM, and the decoder of *Autoencoder*2 were connected sequentially, forming a structure of encoder–LSTM–decoder.

As in the training process in Autoencoder, the loss function uses the Euclidean distance and stops learning when the validation loss does not decrease. After training, the encoder, decoder, and LSTM will be concatenated into a structure of encoder–LSTM–decoder. The simulated point cloud data of five consecutive frames can be transformed into global features through the encoder, and then, these global features were used to predict the global features of the next frame of real point clouds by LSTM, which can be decoded into real point cloud data by the decoder.

TABLE 2. CD [mm].

	Simulation	Simulation + Correction
Average	15.89	**7.81**
Max	34.17	19.25
Min	6.76	4.48
Standard deviation	5.19	1.90

TABLE 3. CD [mm].

	Simulation	Simulation + Correction
Average	18.64	**10.59**
Max	41.18	26.00
Min	7.93	4.48
Standard deviation	6.24	4.00

TABLE 4. CD [mm].

	Simulation	Simulation + Correction
Average	17.23	**15.94**
Max	29.64	26.57
Min	10.13	9.83
Standard deviation	3.57	2.50

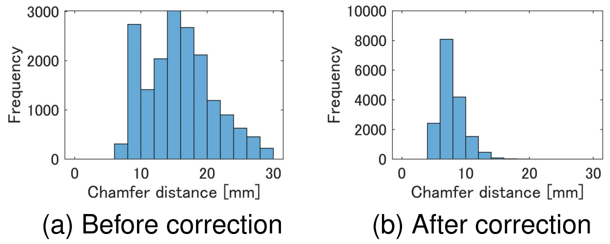


FIGURE 11. CD frequency of basic-type actuator. (a) CD between simulated and real point clouds. (b) CD between corrected simulated and real point clouds.

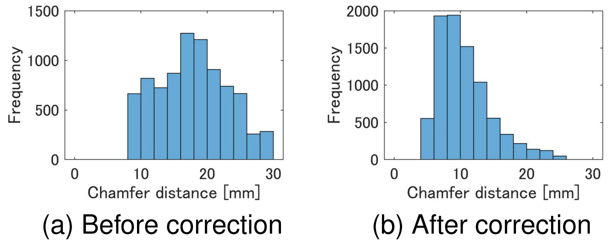


FIGURE 12. CD frequency of side-type actuator. (a) CD between simulated and real point clouds. (b) CD between corrected simulated and real point clouds.

D. ESTIMATION OF PREDICTED POINT CLOUD

The trained encoder–LSTM–decoder network was used to predict real point clouds from the test set. The test set was not used in network training. The basic and side structures were learned by the neural network, whereas the vertical type was a completely unfamiliar robot structure in the evaluation period. The performance of the correction system can be seen in the learned (basic type) and unlearned robot (vertical type) structures. The comparison of the simulation and corrected point clouds in terms of CD is shown in TABLE 2 (basic type), TABLE 3 (side type), and TABLE 4 (vertical type). There is a decrease in CD across all types of executors in values of average, max, min, and standard deviation. In the base and side types, the average CD decreased by 50.85% and 43.19%, respectively, whereas the average CD of the vertical type only decreased by 7.49%.

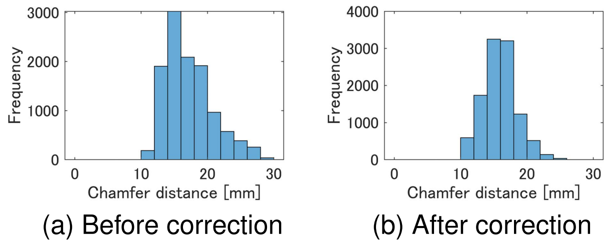


FIGURE 13. CD frequency of vertical-type actuator. (a) CD between simulated and real point clouds. (b) CD between corrected simulated and real point clouds.

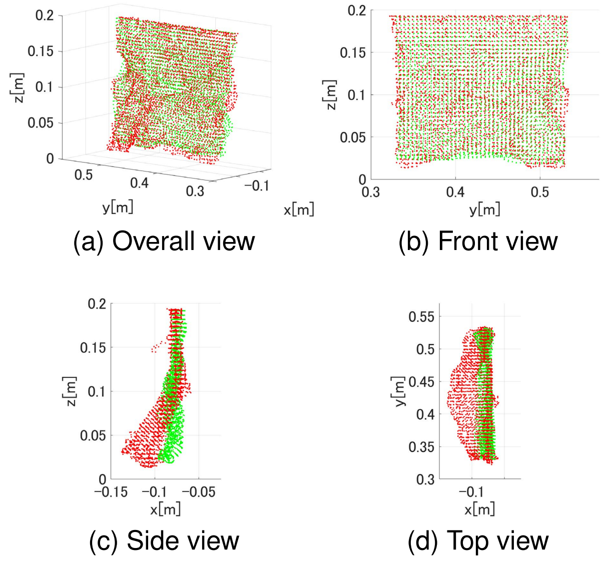


FIGURE 14. Simulated point cloud of basic-type actuator (CD: 18.33 mm). Green point: simulated point cloud. Red point: real point cloud.

In addition, Figs. 11 - 13 shows the CD frequency distribution for each type of actuator before and after correction. Comparing the simulations without the correction system (Fig. 11(a), Fig. 12(a)) and the simulations with the correction system (Fig. 11(b), Fig. 12(b)) in basic type and side types, the CD of the point cloud decreased after correction; i.e., the overall value of CD decreased. Therefore, based on our neural network, the correction system can learn high-dimensional features of muscle structure in the actuator, and it is possible to simulate deformations close to those of the actual actuator.

Comparing the simulation of the vertical type (Fig. 13(a)) and the simulations with the correction system (Fig. 13(b)), we find that CD, greater than 17, is improved after correction. Although the vertical-type actuator has an untrained structure, the correction system can reduce CD. Therefore, even if the actuator structure is untrained, deformation can be simulated to some extent using the correction system.

Figure 14 compares the original simulated point cloud with the real point cloud of the basic-type actuator in the overall, front, right-side, and top views sequentially, and Fig. 15 shows the corrected point cloud (CD decreased by 12.41 mm) of the basic-type actuator. The actual point cloud and the point cloud obtained using the correction system are almost the same. Moreover, Fig. 16 depicts the original simulated

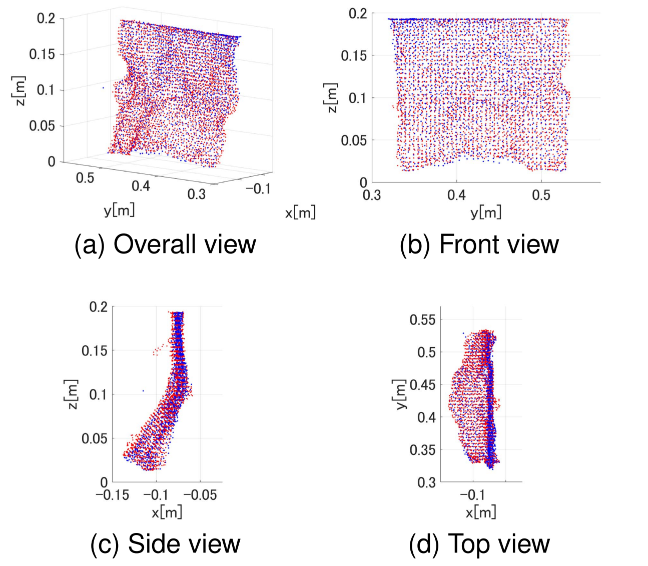


FIGURE 15. Corrected point cloud of basic-type actuator (CD: 5.92 mm). Blue point: corrected point cloud. Red point: real point cloud.

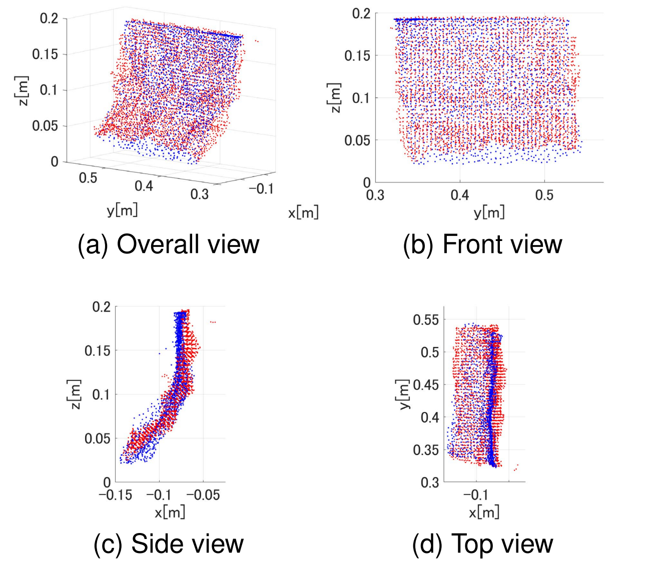


FIGURE 17. Corrected point cloud of vertical-type actuator (CD: 14.45 mm). Blue point: corrected point cloud. Red point: real point cloud.

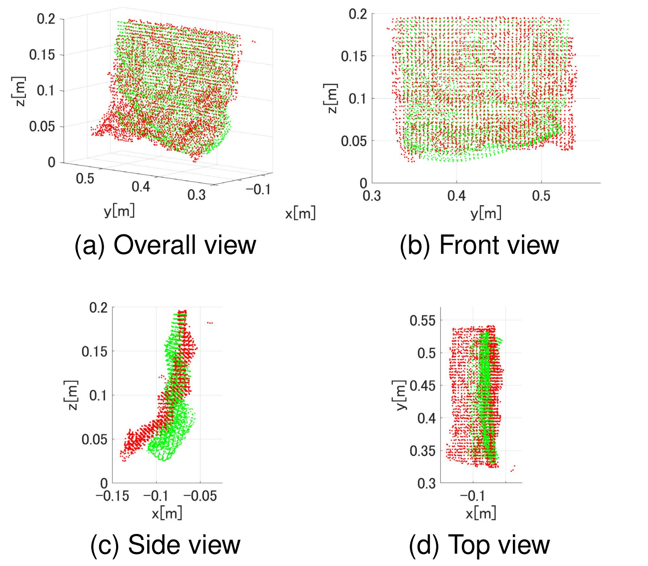


FIGURE 16. Simulated point cloud of vertical-type actuator (CD: 20.15 mm). Green point: simulated point cloud. Red point: real point cloud.

point cloud of the vertical-type actuator, and Fig. 17 shows the corrected point cloud (CD decreased by 4.37 mm) of the basic-type actuator. The overall features of the point clouds after correction are accurate.

To intuitively see which regions of the point cloud were corrected, we selected four area examples (see Fig. 18) from Figs. 14 - 16 (two from the basic type and two from the vertical type), and compared the deformation of the actuator in the selected four areas before and after correction in Figs. 19 - 22.

Figures 19 - 22 show that the majority of the differences between the simulated and real point clouds in the basic type are significantly corrected, whereas the differences between the simulated and real point clouds in the vertical type are mainly corrected, but new errors were generated.

VI. DISCUSSION

We compared the trained and untrained data with the simulation results only and the simulation results with the correction system. The correction system can correct nonlinear mechanical actions difficult to simulate using the PBD-based simulator. In this study, the learned structures performed well in the evaluation stage, and the unlearned structures also improved. However, there is still room for improvement. For example, the neural network has not sufficiently learned about the deformation of unlearned muscle structures, and the correction system's accuracy can still be improved. Moreover, this study employed deep learning to learn time-series point cloud features to compensate for the lack of accuracy of PBD modeling on flexible objects and to facilitate the control of flexible robots in the future.

In addition, there are only three types of artificial muscle distributions (the types of basic, side, and cross) used in the training set in this study. From Table 2 and 3 and Fig. 11 and 12, for the same muscle type, more training data can improve the correction ability of the neural network. However, for unseen data, we tentatively put forward that adding more muscle distribution types to the training set can improve the generalization ability of the neural network, that is, the neural network's ability to correct the simulated point cloud of unlearned muscle structures will be improved.

The structure of the constructed correction system was developed using the PointNet autoencoder and LSTM. In future research, more neural network structures can be used to solve this problem. For example, the attention mechanism. The way a neural network learns is similar to how the human brain thinks, and there may be an advanced framework for extracting point cloud features in the future. Recently, various

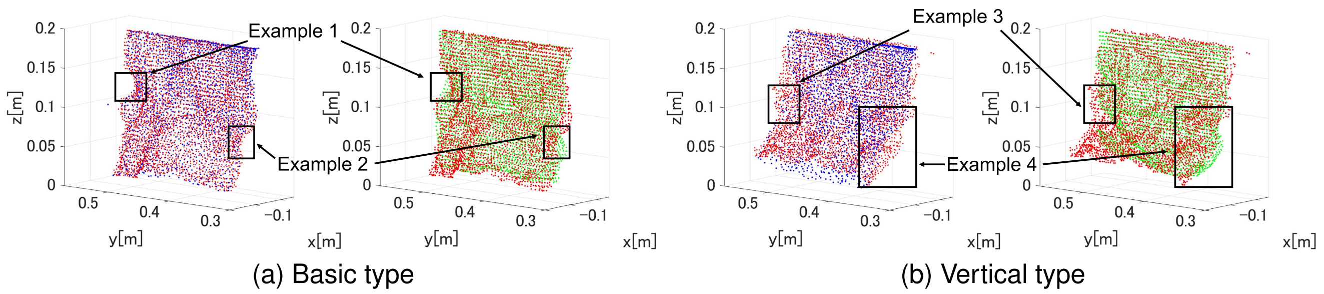


FIGURE 18. The selected areas of each example. Green point: simulated point cloud. Blue point: corrected point cloud. Red point: real point cloud.

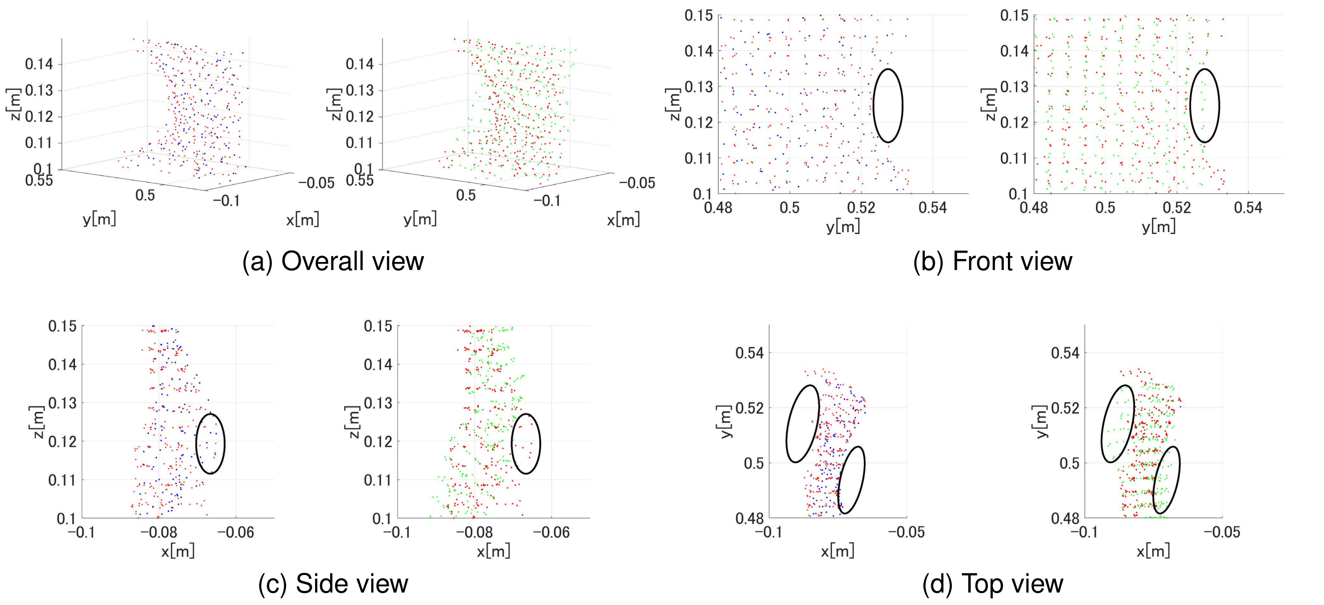


FIGURE 19. Example 1. Green point: simulated point cloud. Blue point: corrected point cloud. Red point: real point cloud. Black circle: the corrected area.

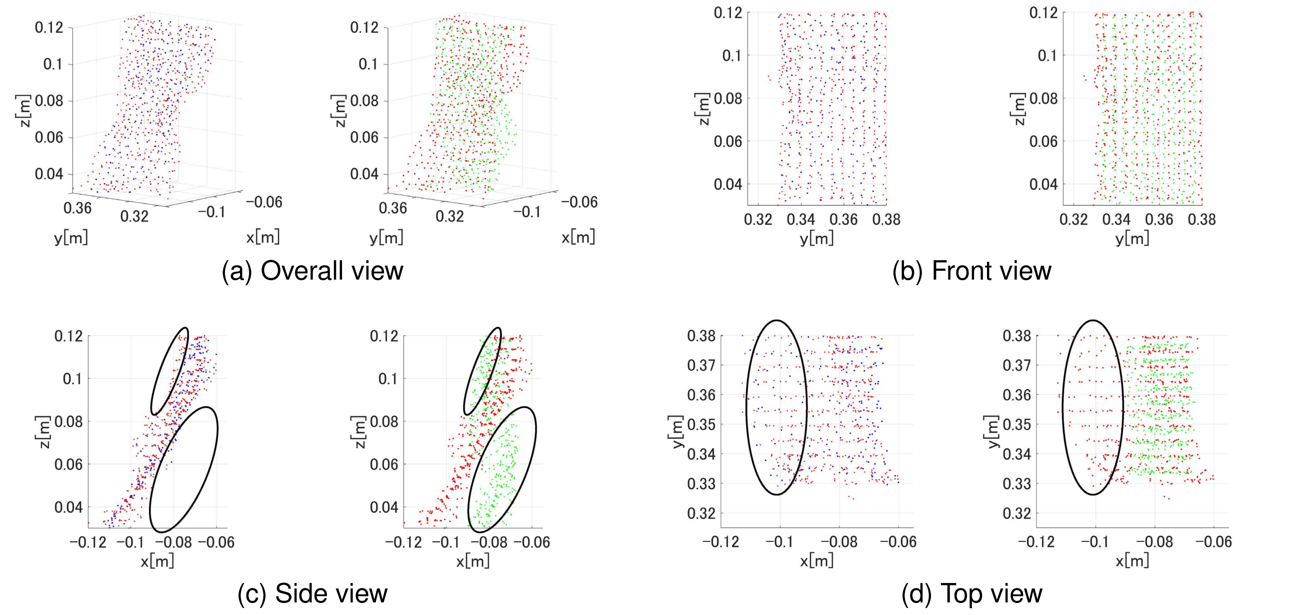


FIGURE 20. Example 2. Green point: simulated point cloud. Blue point: corrected point cloud. Red point: real point cloud. Black circle: the corrected area.

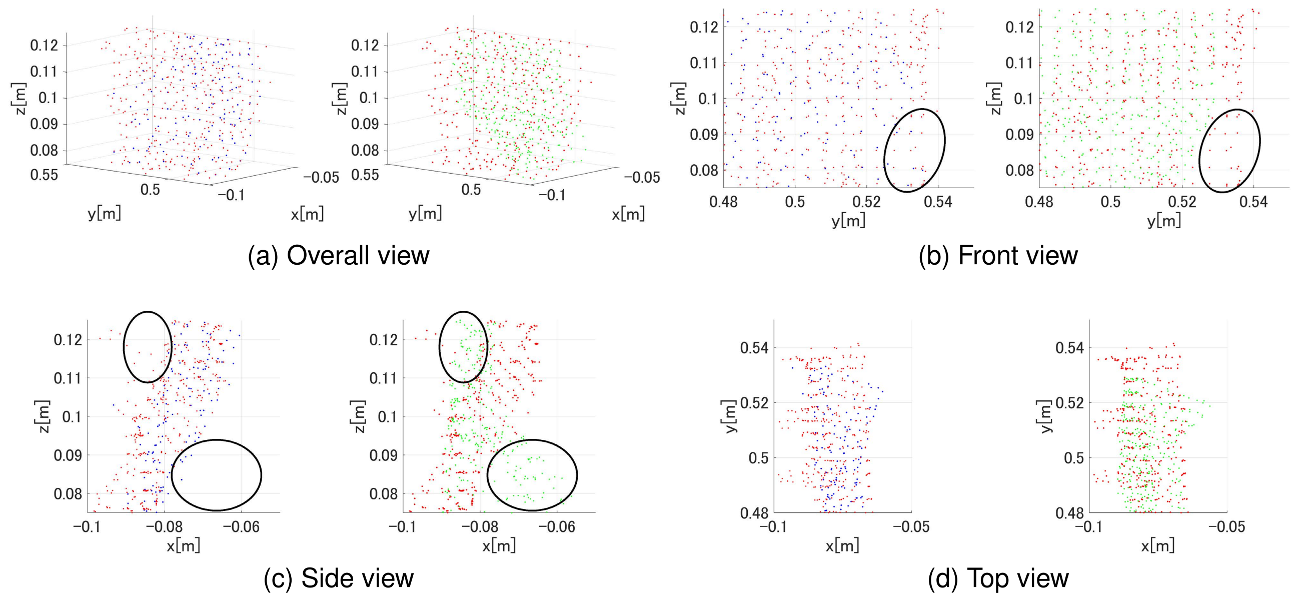


FIGURE 21. Example 3. Green point: simulated point cloud. Blue point: corrected point cloud. Red point: real point cloud. Black circle: the corrected area.

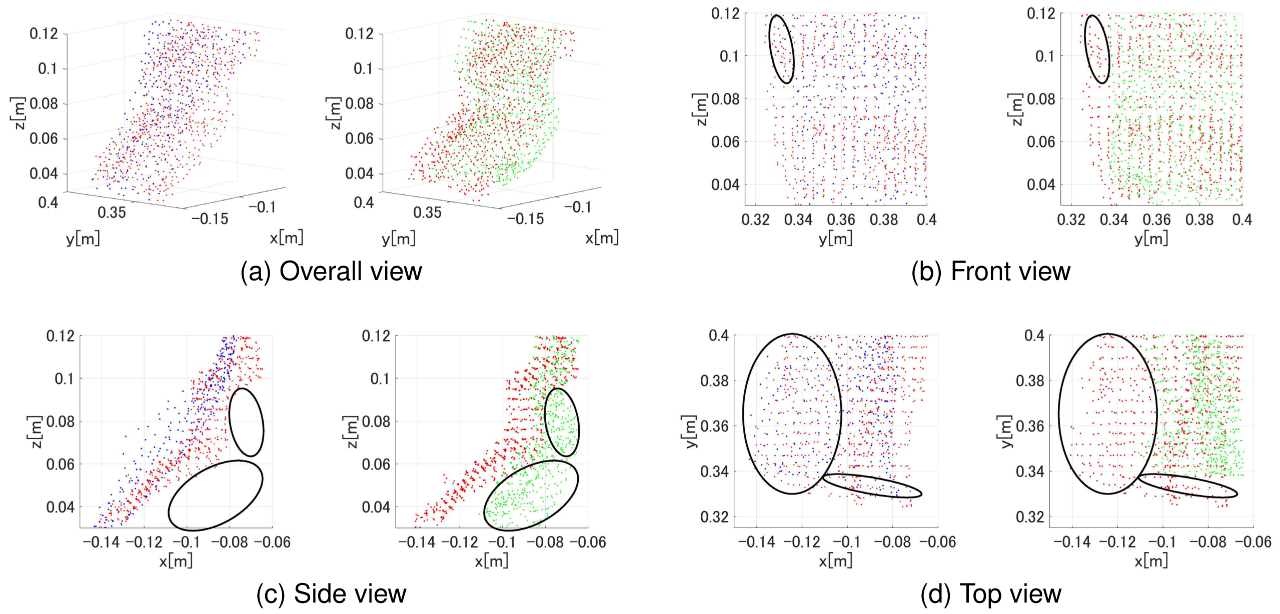


FIGURE 22. Example 4. Green point: simulated point cloud. Blue point: corrected point cloud. Red point: real point cloud. Black circle: the corrected area.

transformers and their variants based on attention mechanisms, such as Point Transformer [57] and Point Cloud Transformer (PCT) [58], have demonstrated significant potential to process point cloud data, achieving high feature recognition performance. Furthermore, note that this study has examined only the time-series point clouds as an input. Adding other information such as the number of artificial muscles as input will also improve the simulation accuracy for untrained structures. Alternatively, simply augmenting the learning data for training the correction system may also increase simulation accuracy, but this may train the neural network deeper rather than wider. In addition, we look forward to adding more muscle structures to the training set, which will enable neural networks to fit higher-dimensional features, and this will have a positive effect on the correction of unknown muscle structures.

This work is intended for use in robot design or control. When the fabric deformation is driven to the expected deformation, many simulated parameters related to muscle and fabric in FEM must be optimized. These muscle parameters include the number of muscles and the length, direction, diameter, and mount point of muscles to be considered. Therefore, FEM-based design is challenging because it requires extensive deformation simulation using various parameters. The constructed correction system can be used to design a structure that can respond to the intended deformations by PBD. In future research, the correction system can facilitate robot designers with rapid real-time simulation and verification of complex wearable devices without using real actuators, owing to the low computational cost. Moreover, we are expecting to evaluate the difference in the specific computational cost between the proposed and FEM method.

VII. CONCLUSION

We proposed a method of modeling real point clouds by correcting simulated point clouds through deep learning to make them similar to real point clouds. As this research has shown, it is feasible to simulate the deformation of known and unknown structures of a fabric-type actuator. Our results are a step toward deep learning-based modeling of soft robotics, which is necessary to correct the inaccuracy of viscoelasticity and nonlinear movements in soft robotic modeling.

There are note-worthy limitations. Although our model simulated learned muscle structures well, the model still needs to learn more muscle structures to improve its generalization ability. Furthermore, more advanced neural network architectures for extracting point cloud features in the future, which merits our continued attention. We hope that our framework can be easily used by other researchers in the field to model the 3D deformation of other soft robot types, anticipating advances in soft robot modeling and control.

REFERENCES

[1] J. Zou, Y. Lin, C. Ji, and H. Yang, "A reconfigurable omnidirectional soft robot based on caterpillar locomotion," *Soft Robot.*, vol. 5, no. 2, pp. 164–174, 2018.

[2] M. Liu, S. Zhu, Y. Huang, Z. Lin, W. Liu, L. Yang, and D. Ge, "A self-healing composite actuator for multifunctional soft robot via photo-welding," *Compos. B, Eng.*, vol. 214, Jun. 2021, Art. no. 108748.

[3] Z. Mao, T. Iizuka, and S. Maeda, "Bidirectional electrohydrodynamic pump with high symmetrical performance and its application to a tube actuator," *Sens. Actuators A, Phys.*, vol. 332, Dec. 2021, Art. no. 113168.

[4] S. Miyagawa, R. Yuasa, H. Nabae, H. Furukawa, and M. Kawakami, "Development of a soft robot with pressure ulcer prevention functions," *J. Robot. Mechtron.*, vol. 34, no. 2, pp. 294–297, 2022.

[5] M. Zhu, T. N. Do, E. Hawkes, and Y. Visell, "Fluidic fabric muscle sheets for wearable and soft robotics," *Soft Robot.*, vol. 7, no. 2, pp. 179–197, Apr. 2020.

[6] B. Ying, R. Z. Chen, R. Zuo, J. Li, and X. Liu, "An anti-freezing, ambient-stable and highly stretchable ionic skin with strong surface adhesion for wearable sensing and soft robotics," *Adv. Funct. Mater.*, vol. 31, no. 42, Oct. 2021, Art. no. 2104665.

[7] J. Guo, C. Xiang, T. Helps, M. Taghavi, and J. Rossiter, "Electroactive textile actuators for wearable and soft robots," in *Proc. IEEE Int. Conf. Soft Robot. (RoboSoft)*, Apr. 2018, pp. 339–343.

[8] M. Tschiersky, E. E. G. Hekman, D. M. Brouwer, J. L. Herder, and K. Suzumori, "A compact McKibben muscle based bending actuator for close-to-body application in assistive wearable robots," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3042–3049, Apr. 2020.

[9] M. Cianchetti, C. Laschi, A. Menciassi, and P. Dario, "Biomedical applications of soft robotics," *Nature Rev. Mater.*, vol. 3, pp. 143–153, May 2018.

[10] L. Wang, G. Peng, W. Yao, S. Biggar, C. Hu, X. Yin, and Y. Fan, "Soft robotics for hand rehabilitation," in *Intelligent Biomechatronics in Neurorehabilitation*, X. Hu, Ed. New York, NY, USA: Academic, 2020, pp. 167–176.

[11] H. Chi, H. Su, W. Liang, and Q. Ren, "Control of a rehabilitation robotic device driven by antagonistic soft actuators," *Actuators*, vol. 10, no. 6, p. 123, Jun. 2021.

[12] S. Koizumi, T.-H. Chang, H. Nabae, G. Endo, K. Suzumori, M. Mita, K. Saitoh, K. Hatakeyama, S. Chida, and Y. Shimada, "Soft robotic gloves with thin McKibben muscles for hand assist and rehabilitation," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Jan. 2020, pp. 93–98.

[13] H. Wang, M. Totaro, and L. Beccai, "Toward perceptive soft robots: Progress and challenges," *Adv. Sci.*, vol. 5, no. 9, Sep. 2018, Art. no. 1800541.

[14] M. Khushi, K. Shaukat, T. M. Alam, I. A. Hameed, S. Uddin, S. Luo, X. Yang, and M. C. Reyes, "A comparative performance analysis of data resampling methods on imbalance medical data," *IEEE Access*, vol. 9, pp. 109960–109975, 2021.

[15] X. Zhang, J. Ding, M. Wu, S. T. C. Wong, H. Van Nguyen, and M. Pan, "Adaptive privacy preserving deep learning algorithms for medical data," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 1169–1178.

[16] H. Song, X.-Y. Han, C. E. Montenegro-Marin, and S. Krishnamoorthy, "Secure prediction and assessment of sports injuries using deep learning based convolutional neural network," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 3, pp. 3399–3410, Mar. 2021.

[17] G. Li, Y. Yang, X. Qu, D. Cao, and K. Li, "A deep learning based image enhancement approach for autonomous driving at night," *Knowl.-Based Syst.*, vol. 213, Feb. 2021, Art. no. 106617.

[18] Y. Deng, T. Zhang, G. Lou, X. Zheng, J. Jin, and Q.-L. Han, "Deep learning-based autonomous driving systems: A survey of attacks and defenses," *IEEE Trans. Ind. Informat.*, vol. 17, no. 12, pp. 7897–7912, Dec. 2021.

[19] M. Hosein Hamian, A. Beikmohammadi, A. Ahmadi, and B. Nasersharif, "Semantic segmentation of autonomous driving images by the combination of deep learning and classical segmentation," in *Proc. 26th Int. Comput. Conf., Comput. Soc. Iran (CSICC)*, Mar. 2021, pp. 1–6.

[20] D.-H. Lee, K.-L. Chen, K.-H. Liou, C.-L. Liu, and J.-L. Liu, "Deep learning and control algorithms of direct perception for autonomous driving," *Appl. Intell.*, vol. 51, no. 1, pp. 237–247, 2021.

[21] D. Kim, S.-H. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee, J. Bae, Y.-L. Park, K.-J. Cho, and S. Jo, "Review of machine learning methods in soft robotics," *PLoS ONE*, vol. 16, no. 2, Feb. 2021, Art. no. e0246102.

[22] Y. Guo and H. Peng, "Full-actuation rolling locomotion with tensegrity robot via deep reinforcement learning," in *Proc. 5th Int. Conf. Robot. Autom. Sci. (ICRAS)*, Jun. 2021, pp. 51–55.

[23] S. Hofer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, C. Karen Liu, J. Peters, S. Song, P. Welinder, and M. White, "Sim2Real in robotics and automation: Applications and challenges," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 2, pp. 398–400, Apr. 2021.

[24] S. Yang, X. He, and B. Zhu, "Learning physical constraints with neural projections," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 5178–5189.

[25] A. Segato, M. D. Marzo, S. Zucchelli, S. Galvan, R. Secoli, and E. De Momi, "Inverse reinforcement learning intra-operative path planning for steerable needle," *IEEE Trans. Biomed. Eng.*, vol. 69, no. 6, pp. 1995–2005, Jun. 2022.

[26] W. Jones, S. Gangapurwala, I. Havoutis, and K. Yoshida, "Towards generating simulated walking motion using position based deep reinforcement learning," in *Proc. Annu. Conf. Towards Auton. Robot. Syst.* Cham, Switzerland: Springer, 2019, pp. 467–470.

[27] R. B. N. Scharff, G. Fang, Y. Tian, J. Wu, J. M. P. Geraedts, and C. C. L. Wang, "Sensing and reconstruction of 3-D deformation on pneumatic soft robots," *IEEE/ASME Trans. Mechatronics*, vol. 26, no. 4, pp. 1877–1885, Aug. 2021.

[28] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.

[29] J. Zhao, X. Chen, X. Dou, Y. Cao, Y. Wang, Y. Cui, and X. Niu, "Automatic classification of medicinal materials based on three-dimensional point cloud and surface spectral information," *Proc. SPIE*, vol. 12168, pp. 384–394, Mar. 2022.

[30] Y. Qian, Q. Xu, Y. Yang, H. Lu, H. Li, X. Feng, and W. Yin, "Classification of Rice seed variety using point cloud data combined with deep learning," *Int. J. Agricult. Biol. Eng.*, vol. 14, no. 5, pp. 206–212, 2021.

[31] O. Majgaonkar, K. Panchal, D. Laefer, M. Stanley, and Y. Zaki, "Assessing LiDAR training data quantities for classification models," *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 46, pp. 101–106, Oct. 2021.

[32] X. Chen, K. Jiang, Y. Zhu, X. Wang, and T. Yun, "Individual tree crown segmentation directly from UAV-borne LiDAR data using the PointNet of deep learning," *Forests*, vol. 12, no. 2, p. 131, Jan. 2021.

[33] J. Liu, W. Xiong, L. Bai, Y. Xia, T. Huang, W. Ouyang, and B. Zhu, "Deep instance segmentation with automotive radar detection points," *IEEE Trans. Intell. Vehicles*, early access, Apr. 22, 2022, doi: 10.1109/TIV.2022.3168899.

[34] Y. Li, W. Wen, T. Miao, S. Wu, Z. Yu, X. Wang, X. Guo, and C. Zhao, "Automatic organ-level point cloud segmentation of maize shoots by integrating high-throughput data acquisition and deep learning," *Comput. Electron. Agricult.*, vol. 193, Feb. 2022, Art. no. 106702.

[35] Z. Yu and K. Liao, "Semantic segmentation of 3-D SAR point clouds by graph method based on PointNet," in *Proc. 11th Int. Conf. Comput. Eng. Netw.* Singapore: Springer, 2022, pp. 408–418.

[36] C. R. Qi. (2019). *PointNet-Autoencoder*. [Online]. Available: https://github.com/charlesq34/pointnet-autoencoder

[37] H. Xie, J. Song, Y. Zhong, and C. Gu, "Kalman filter finite element method for real-time soft tissue modeling," *IEEE Access*, vol. 8, pp. 53471–53483, 2020.

[38] C. Duriez and T. Bieze, "Soft robot modeling, simulation and control in real-time," in *Soft Robotics: Trends, Applications and Challenges*. Cham, Switzerland: Springer, 2017, pp. 103–109.

[39] Y. Dang, M. Stommel, L. K. Cheng, and W. Xu, "A soft ring-shaped actuator for radial contracting deformation: Design and modeling," *Soft Robot.*, vol. 6, no. 4, pp. 444–454, Aug. 2019.

[40] Z. Qin, Y. Tai, C. Xia, J. Peng, X. Huang, Z. Chen, Q. Li, and J. Shi, "Towards virtual VATS, face, and construct evaluation for peg transfer training of box, VR, AR, and MR trainer," *J. Healthcare Eng.*, vol. 2019, pp. 1–10, Jan. 2019.

[41] F. Liu, Z. Li, Y. Han, J. Lu, F. Richter, and M. C. Yip, "Real-to-sim registration of deformable soft tissue with position-based dynamics for surgical robot autonomy," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 12328–12334.

[42] F. Liu, M. Li, J. Lu, E. Su, and M. C. Yip, "Parameter identification and motion control for articulated rigid body robots using differentiable position-based dynamics," 2022, *arXiv:2201.05753*.

[43] P. Tripicchio, S. D'Avella, and E. Ruffaldi, "Real-time numerical simulation for accurate soft tissues modeling during haptic interaction," *Actuators*, vol. 11, no. 1, p. 17, Jan. 2022.

[44] Y. Funabora, "Flexible fabric actuator realizing 3D movements like human body surface for wearable devices," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 6992–6997.

[45] W. Yu, J. Gonzalez, and X. Li, "Fast training of deep LSTM networks with guaranteed stability for nonlinear system modeling," *Neurocomputing*, vol. 422, pp. 85–94, Jan. 2021.

[46] Y. Fan, K. Xu, H. Wu, Y. Zheng, and B. Tao, "Spatiotemporal modeling for nonlinear distributed thermal processes based on KL decomposition, MLP and LSTM network," *IEEE Access*, vol. 8, pp. 25111–25121, 2020.

[47] T. Li, T. Wu, and Z. Liu, "Nonlinear unsteady bridge aerodynamics: Reduced-order modeling based on deep LSTM networks," *J. Wind Eng. Ind. Aerodyn.*, vol. 198, Mar. 2020, Art. no. 104116.

[48] S. Kurumaya, H. Nabae, G. Endo, and K. Suzumori, "Design of thin McKibben muscle and multifilament structure," *Sens. Actuator A, Phys.*, vol. 261, pp. 66–74, Jul. 2017.

[49] M. Takaoka, K. Suzumori, S. Wakimoto, S. Iijima, and T. Tokumiya, "Fabrication of thin McKibben artificial muscle with various design parameters and their experimental evaluations," in *Proc. 5th Int. Conf. Manuf., Mach. Design Tribol.*, 2013, p. 82.

[50] N. Gameworks. (2018). *NVIDIA FleX*. [Online]. Available: https://docs.nvidia.com/gameworks/content/artisttools/Flex/index.html

[51] M. Macklin, M. Müller, N. Chentanez, and T. Y. Kim, "Unified particle physics for real-time applications," *ACM Trans. Graph.*, vol. 33, p. 153, Jul. 2014.

[52] A. Ohno, Y. Yamamoto, M. Oguro, and K. Suzumori, "Comparison in characteristics of textile woven by thin pneumatic artificial muscle," in *Proc. Abstracts Int. Conf. Adv. Mechatronics, Toward Evol. Fusion Mechatronics*, 2015, pp. 43–44.

[53] S Muscle. (2022) *Technical Data*. [Online]. Available: https://www.s-muscle.com/%E6%8A%80%E8%A1%93%E8%B3%87%E6%96%99/

[54] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," *Proc. SPIE*, vol. 1611, pp. 586–606, Apr. 1992.

[55] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 605–613.

[56] A. Mikolajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *Proc. Int. Interdiscipl. PhD Workshop (IIPhDW)*, May 2018, pp. 117–122.

[57] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun, "Point transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 16259–16268.

[58] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "PCT: Point cloud transformer," *Comput. Vis. Media*, vol. 7, no. 2, pp. 187–199, Jun. 2021.

YANHONG PENG (Member, IEEE) received the B.E. degree in mechatronic engineering from Beijing Jiaotong University, China, and the University of Wollongong, Australia, in 2019, and the M.E. degree in automotive engineering from the Department of Mechanical System Engineering, Nagoya University, Japan, in 2021, where he is currently pursuing the Ph.D. degree in information and communication engineering. His research interests include soft robotics and machine learning.

HIROKI YAMAGUCHI received the B.E. and M.E. degrees in electrical and electronic engineering and information engineering from Nagoya University, Japan, in 2019 and 2021, respectively. He is with Mitsubishi Electric. His interest includes soft robotics.

YUKI FUNABORA (Member, IEEE) received the B.E., M.E., and Ph.D. degrees in electrical engineering and computer science from Nagoya University (NU), Japan, in 2007, 2009, and 2012, respectively. In 2012, he was a Postdoctoral Researcher at the RIKEN Advanced Science Institute. Since 2013, he has been an Associate Professor with NU. His research interests include human-cooperative robots, soft robotics, intelligent control, soft computing, and system design.

SHINJI DOKI (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in electronic-mechanical engineering from Nagoya University (NU), Japan, in 1990, 1992, and 1995, respectively. Since 2012, he has been a Professor with NU. His research interests include the control, modeling, and signal/information processing and its application for motor drive systems. He was a recipient of the IEEE IECON '92 Best Paper Award, as well as the paper awards from FANUC FA, Robot Foundation, and the Institute of Electrical Engineers of Japan.