

Harnessing Deep Learning of Point Clouds for Morphology Mimicking of Universal 3D Shape-Morphing Devices

Jue Wang, Dhirodaatto Sarkar, Jiaqi Suo, and Alex Chortos*

Shape-morphing devices, a crucial branch in soft robotics, hold significant application value in areas like human–machine interfaces, biomimetic robotics, and tools for biological systems. To achieve 3D programmable shape morphing (PSM), the deployment of array-based actuators is essential. However, a critical knowledge gap in 3D PSM is controlling the complex systems formed by these soft actuator arrays to mimic the morphology of the target shapes. This study, for the first time, represents the configuration of shape-morphing devices using point cloud data and employing deep learning to map these configurations to control inputs. Shape Morphing Net (SMNet), a method that realizes the regression from point cloud to high-dimensional control input vectors, is proposed. It has been applied to 3D PSM devices with three different actuator mechanisms, demonstrating its universal applicability to inversely reproduce the target shapes. Further, applied to previous 2D PSM devices, SMNet significantly enhances control precision from 82.23% to 97.68%. In the demonstrations of morphology mimicking, 3D PSM devices successfully replicate arbitrary target shapes obtained either through 3D scanning of physical objects or via 3D modeling software. The results show that within the deformable range of 3D PSM devices, accurate reproduction of the desired shapes is achievable.

1. Introduction

Shape-morphing devices, a pivotal subset of soft robotics, aspire to achieve programmable, controllable, and reversible transformations reminiscent of biological systems such as octopi and growing plants.^[1,2] They exhibit potential in realms such as human–machine interfaces for augmented and virtual reality

devices,^[3] haptic technology,^[4] optical and acoustic metamaterials,^[5] and devices for manipulating biology.^[6–8]

The most commonly investigated shape-morphing devices morph between two distinct shapes, driven by the material design or structural configurations.^[9–15] We refer to these devices as pattern-to-pattern shape morphing. Many emerging applications require controllable and reversible transformations, which has stimulated the development of devices that can transform their structure on demand, which we refer to as programmable shape morphing (PSM).^[11,16–21] Such devices consist of an array of actuators, enabling a singular device to transform into various configurations as necessitated.

Early PSM systems were composed of arrays of solid linear actuators that could reproduce surfaces on demand.^[22,23] Since all actuators were mechanically decoupled, the control algorithms were relatively simple. However, the bulky and cumbersome nature of the linear actuators and their control equipment^[24,25] limited their applications.

Recent advancements in materials and fabrication techniques have led to the emergence of flexible actuators,^[26] substantially reducing the size of actuator arrays and enabling the creation of entirely continuous surfaces.^[20,21] This has revitalized interest in PSM. Given the intricate coupling that exists between the deformations of different actuators, the control algorithm plays an important role in continuous PSM devices. For the works whose deformation is generated by rod-shaped actuators connected by points,^[17,20] the construction of analytical models is feasible, albeit necessitating certain simplifications. However, the reliance on simplifying assumptions restricts the design freedom of devices. Continuum actuators that actuate throughout their surface or volume^[16,21,27] present a more generalized approach to deformation and mimic the continuous nature of biological systems. Yet, the pronounced geometric coupling inherent to arrays of continuous actuators presents significant hurdles for traditional analytical models. Machine learning has recently emerged as a strategy to achieve model-free control of these complex systems.^[21,27] The approach uses finite element modeling to generate the dataset for machine learning. Finite element modeling is effective for forward simulations, which calculate the expected deformation of a PSM based on given control inputs. A dataset is created by performing thousands of these forward simulations with random control inputs. A machine

J. Wang, D. Sarkar, A. Chortos
School of Mechanical Engineering
Purdue University
610 Purdue Mall, West Lafayette 47907, Indiana, USA
E-mail: achortos@purdue.edu

J. Suo
School of Construction Management Technology
Purdue University
610 Purdue Mall, West Lafayette 47907, Indiana, USA

 The ORCID identification number(s) for the author(s) of this article can be found under <https://doi.org/10.1002/aisy.202400550>.

© 2024 The Author(s). Advanced Intelligent Systems published by Wiley-VCH GmbH. This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

DOI: 10.1002/aisy.202400550

learning model then correlates the deformed state of the actuator with the control inputs. The machine learning model can then be executed to determine the control inputs that are required to achieve a target deformation (inverse control).

Currently, research on shape-morphing devices has predominantly centered around 2D arrays of actuators that are capable of transforming into 3D surfaces^[20,21,28] due to the availability of 2D fabrication approaches. With the increasing development of 3D fabrication techniques,^[29,30] 3D arrays of actuators have become viable. However, for controlling these 3D arrays of coupled actuators, their complexity increases exponentially compared to the existing 2D arrays. Consequently, the control algorithms are the crucial knowledge gap in achieving 3D shape morphing.

For soft robots with serial structures, it is possible to use parametric representation of the deformed geometry to accomplish machine learning-based model-free control.^[31,32] For shape-morphing devices with 2D arrays of actuators, 1D data are sufficient to describe their deformation.^[17,20,21,27] However, for the highly complex 3D arrays of actuators, neither parametric representation nor 1D data can adequately capture their intricate deformations in 3D space. Therefore, point cloud data are the most direct and specific methods for representing these deformed geometries.

The versatility of point clouds is reflected by their widespread adoption in fields that rely on 3D representations, including the construction industry,^[33] autonomous navigation,^[34] computer vision,^[35] and robot sensing.^[36,37] Building upon this, the application of machine learning to point cloud data has opened new frontiers. Machine learning tasks for point clouds primarily encompass segmentation,^[38,39] classification,^[40,41] and reconstruction.^[42,43] These methodologies have found prolific applications in areas such as robotic sensing, autonomous driving, and geoscience. While regression tasks with point clouds are less prevalent, they have garnered attention in niche domains including forest biomass estimation,^[44] reconstruction of deformable objects,^[45] and hand pose recognition.^[46]

Morphology mimicking of 3D shape morphing, which involves controlling their deformation based on inputted target shapes, can be described as a mapping between the target shape, expressed in the point cloud, and the control inputs. It can be considered a typical point cloud regression task.

Therefore, in this study, we present a universal approach to inversely control 3D PSM devices with different actuation principles. For the first time, we express the deformation of shape-morphing devices using 3D point cloud data and propose a novel machine learning model to correlate this representation with the input vector that represents the control inputs to an array of actuators (**Figure 1A**). This methodology can determine the control signals necessary to achieve an input target shape, which can then be used to reproduce the desired configuration. The process can be applied to any actuating system for which a finite element model can be created, thus providing a universal control method for 3D PSM devices with any deformation principle.

Our training data are derived from finite element analysis (FEA) simulations. We initiated our research by establishing a simulation model for the 3D PSM devices. Subsequently, large numbers of FEA simulations were run with random control inputs to procure the deformed 3D point cloud data. To correlate the deformed state (point cloud data) with the control inputs (continuous high-dimensional vectors), we introduce a new training architecture, named Shape Morphing Net (SMNet), that is uniquely well suited for this task. In this model, we initially employ Kernel Point Convolution (KPConv) to extract hidden features from the point cloud, which are then merged with the original data to enhance it. Subsequently, the augmented data are processed through PointNet++ to transform it into feature vectors, which is then used for regression analysis against the control signals of actuator arrays (the output of this model). This model not only achieves the morphology mimicking for 3D PSM devices but also significantly enhances the control precision of 2D PSM device. To verify its universality, we applied it to three 3D PSM devices based on different deformation

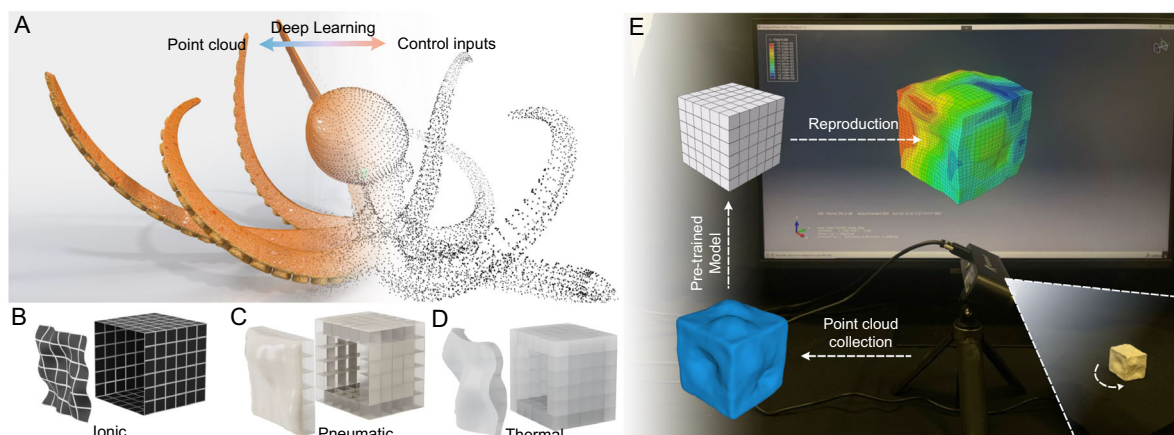


Figure 1. A universal method for controlling 3D PSM devices by mapping the point cloud of the deformed configuration with the control inputs of devices. A) Utilizing the point cloud to express the configurations of shape-morphing devices, using an octopus to symbolize a shape-morphing device. B) The rendering for 3D PSM devices based on ionic actuator arrays. C) The rendering for 3D PSM devices based on pneumatic actuator arrays. D) The rendering for 3D PSM devices based on thermal actuator arrays. E) The demonstration procedures by which 3D PSM devices reproduce the shape of physical objects in simulations.

principles, and the results achieved high-precision inverse control in all cases (Figure 1B–D). To validate the efficacy of our method in mimicking the morphology of real-world objects, we captured point cloud data of a physical object using a 3D scanner. The data were processed and input into a pretrained model, generating a control input array for 3D PSM devices. Subsequently, using these control inputs, all mechanisms were able to accurately replicate the object's shape (Figure 1E). Additionally, we demonstrated that our proposed method is equally capable of reproducing the target shapes derived from more complex virtual 3D models.

2. Results

2.1. Data Collection and Preprocessing of Point Cloud Data

In this study, we employed four datasets for training. One dataset originates from our group's prior work,^[21] on continuous actuator arrays made of ionic actuators. The other three datasets are created from 3D PSM devices with three different actuation mechanisms. All data are derived from FEA simulations, with

detailed simulation processes delineated in the Section 4. When generating the training data from FEA simulations, we needed a large amount of data that are not directly correlated with each other. Therefore, all the control signals were generated randomly and were uniformly distributed. These control signals were then sequentially applied to the actuators in the simulation to produce different deformation data, which were used as the training set. Then, for postsimulation, we extracted the “XYZ” displacement data for nodes, encompassing two segments: pre-deformation node positions and postdeformation node displacements. Summing them yielded the 3D point cloud data for the postdeformation state. Given the variability in mesh and 3D models across simulations, we implemented a standardized preprocessing routine for the point clouds. The process of reproducing a 3D structure begins with a 3D scan (Figure 1E) that provides the external points of the structure. Consequently, the first step of preprocessing the data from FEA simulations involved eliminating internal points (Figure 2A). Owing to the heterogeneity introduced by the tetrahedral mesh, the point cloud distribution lacked uniformity. We first employed a grid average method for downsampling, which ensured uniform point distribution. However, the resultant number of points

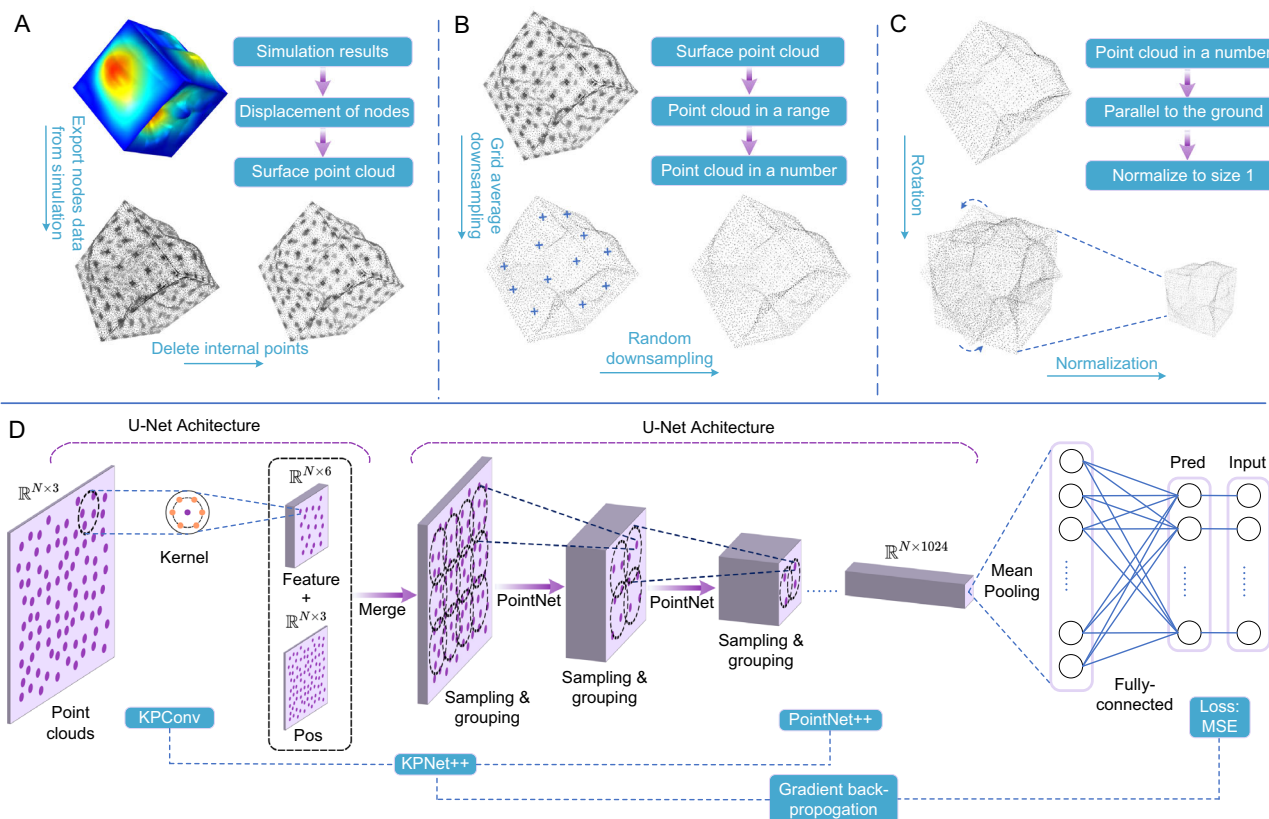


Figure 2. The framework of mapping point cloud from simulation results with the control inputs by using SMNet. A) The procedures of extracting point cloud data from simulations. B) The downsampling strategy for point cloud data: including grid average downsampling to avoid the point concentration and random downsampling to ensure the number of points is the same. C) The point cloud rotation and normalization for training requirements. D) The architecture of SMNet for regression problems.

varied, prompting us to further employ a random downsampling technique, adjusting all point cloud counts to “N” (the smallest count postgrid average downsampling within the dataset, ensuring minimal random deletions) (Figure 2B). The detailed explanation of the point cloud downsampling is shown in Section S1, Supporting Information. Subsequently, using the predeformation node positions, we centered and rotated the point cloud. This ensured that for 2D models, the point (0, 0) was centered within a square aligned parallel to the x - y plane, and for 3D models, the point (0, 0, 0) was central within a cube, with each face aligned parallel to the x - y , y - z , or x - z planes. Finally, we normalized the postdeformation point cloud data, constraining its range between -0.5 and 0.5 (Figure 2C).

2.2. Architecture of SMNet

SMNet finds the regression between the control inputs of the structure and the resulting deformed geometry, which is represented as a point cloud. The first step of model training consists of processing the point cloud data $\mathcal{P} \in \mathbb{R}^{N \times 3}$ through a KPConv layer to identify the hidden features of the point cloud (Figure 2D). We construct a 1D feature vector composed entirely of ones $\{\mathcal{F} \in \mathbb{R}^{N \times 1} | \mathcal{F}_{in} = 1\}$, which serves as an initial feature for convolution with the point cloud. The whole KPConv part is built upon the foundational U-Net architecture, which comprises a sequence of encoding (downsampling) layers followed by a symmetrical set of decoding (upsampling) layers. Crucial to this structure are skip connections that directly link layers from the encoder to their counterpart layers in the decoder, ensuring the preservation and fusion of multiscale features. Within its intermediate layers, unlike traditional convolutional operations that operate on standardized grids, KPConv is distinctively equipped with deformable kernels, allowing the convolutional kernels to adapt to more complex and varying geometric patterns, thereby enhancing the model's capacity to represent intricate spatial relationships in the data. This convolution process yields point cloud features as $\mathcal{F}_{out} \in \mathbb{R}^{N \times 6}$.

Subsequent to this, we combine the newly generated features $\mathcal{F}_{out} \in \mathbb{R}^{N \times 6}$ with the original point cloud $\mathcal{P} \in \mathbb{R}^{N \times 3}$ to a new dataset $\mathcal{P}_f \in \mathbb{R}^{N \times 9}$ and then input to the advanced PointNet++ architecture. The base architecture of PointNet++ is also built upon U-Net architecture similar to the KPConv. In each encoding and decoding layer, there is a sampling process by using spheres to reorganize the point set followed by a grouping process to integrate the centroid points of each sphere with the points in the neighborhood of centroid points. Subsequently, the aggregated data pass through a mini-PointNet network consisting of convolution, normalization, and ReLU activations for each layer.

After progressing through the PointNet++ framework, the output feature $\mathcal{F}'_{out} \in \mathbb{R}^{N \times 1024}$ is subjected to average pooling. Here, given that the point cloud data are extracted from simulation results, it is inherently devoid of noise. Moreover, the deformations present are rather continuous, lacking in pronounced local detail features. Consequently, we opt for average pooling over max pooling. Then, fully connected layers are used to connect the feature vector from pooling to the target 152-dimensional (216 in 3D ionic case) output vector with the

ReLU as activation function of each layer. Since the model's predictions are continuous number from -1 to 1 , the mean squared error (MSE) is employed as the pivotal loss function. The gradients of this loss with respect to the model parameters are computed and used to update the model's weights (The detailed description of SMNet is introduced in the Section 4.).

2.3. Model Performance of Ionic 2D PSM and 3D PSM Devices

For the 2D ionic PSM device, it is integrated with a 6×6 array of square-shaped ionic actuators as shown in Figure 3A. In experimental systems,^[21] each of these actuators can be independently controlled. Intriguingly, these ionic actuators exhibit bidirectional deformation; they bend downward upon the application of a positive voltage to the upper electrode and vice versa. The data employed herein derive from the simulation results in Abaqus, as previously published by our group. In both simulations and physical experiments of the ionic 2D PSM, the central point of the square is held fixed, a strategy employed to maximize deformation amplitude. This fixation method, however, induces a pronounced x - y shift in the postdeformation point cloud, making the reliance solely on z -axis data inadequate for learning. For detailed simulation configurations and data extraction protocols, readers are directed to our earlier work.^[21] Concerning this PSM, we have amassed a training set of 5000 samples and a test set comprising 100 samples. To shed light on model performance and error distribution, an error map of the 100 ground-truth input vectors for PSM control with the model's predicted vectors was executed. These disparities were visually conveyed through a 6×6 color map, where each section indicates the input error of the corresponding actuator on the actual PSM, as depicted in Figure 3B. In subsequent analyses, the predicted vectors were injected into the simulation model to recreate the PSM shapes. A comparative assessment between the point cloud data of these reproduced shapes and the test set unveiled the error cloud map, showcased in Figure 3C. This error cloud map, with a resolution of 30×30 , aggregates and averages the error within each grid, color-coded to represent varying magnitudes of discrepancies. As evident from Figure 3C, due to the centrality of the fixed point, discrepancies predominantly amass in the upper right and lower left quadrants, while the central region showcases minimal error.

It is worth mentioning that beyond the SMNet model proposed in this study, we have also adapted prevalent point cloud segmentation and classification architectures like 3DCNN,^[47,48] PointNet,^[49] PointNet++,^[50] KPConv,^[51] and RSConv^[52] for our regression task, serving as benchmarks. Unlike these models, SMNet demonstrates a distinctive advantage in handling shape-morphing 3D point cloud data. This data, derived from simulation nodes, consists solely of coordinate information and is stable with no noise. As this 3D data lacks additional features, SMNet initially employs KPConv to acquire local shape features of the point cloud beyond coordinates. Subsequently, it utilizes PointNet++ to integrate and learn from both coordinates and local shape features, making it particularly effective for this specific application. Upon completing training, the coefficient of determination, R2 score, gleaned from the test set, was selected as the performance metric. The bar figure of models' R2 scores is portrayed in Figure 3D, and the multilayer perceptron (MLP)

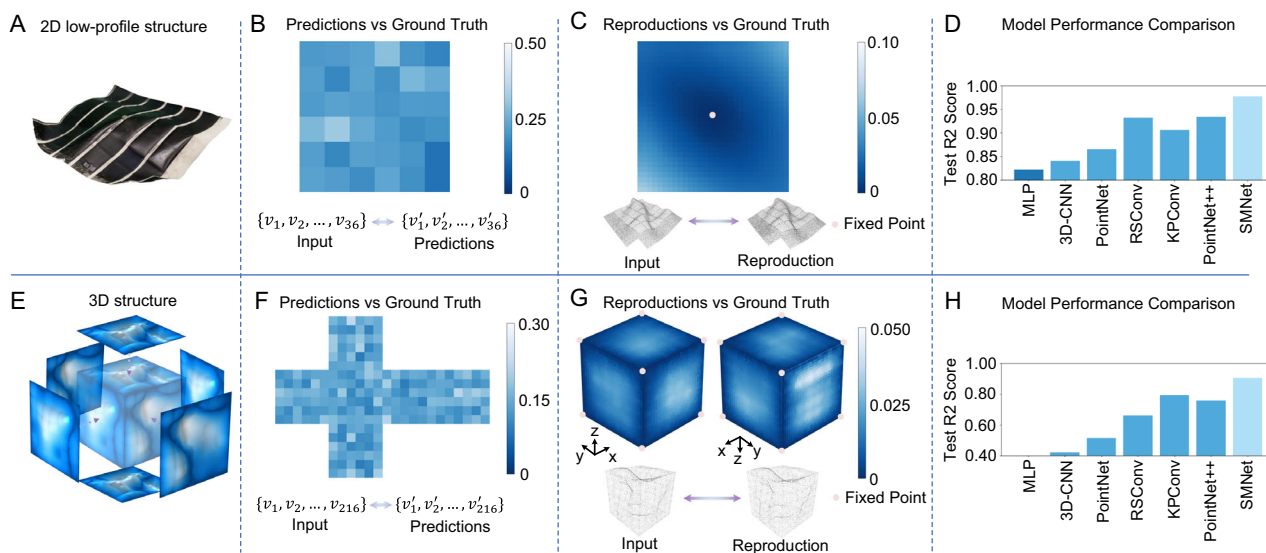


Figure 3. The model performance of ionic 2D PSM and 3D PSM device. A) The physical image of ionic 2D PSM device proposed in ref. [21]. B) The error map between model predictions and the ground-truth control input vectors of ionic 2D PSM device. C) The error map of ionic 2D PSM device showcasing the point cloud of reproduced shapes with the ground-truth point cloud. D) The comparison of R2 score across various training models for ionic 2D PSM device. E) The exploded image of ionic 3D PSM device assembled by six pieces of ionic 2D PSM device. F) The unfolded error map between model predictions and the ground-truth control input vectors of ionic 3D PSM device. G) The 3D error map of ionic 2D PSM device showcasing the point cloud of reproduced shapes with the ground-truth point cloud. There are two angles of view to show the entire six surfaces of the cube. H) The comparison of R2 score across various training models for ionic 3D PSM device.

model and SMNet model are highlighted (Figure 3D). In previous endeavors where merely z-displacement was employed—eschewing 3D point cloud data—the MLP achieved a modest accuracy of 0.8223.^[21,27] A paradigm shift to utilizing point cloud data saw every model subjected to five independent training sessions. The resultant mean accuracies were as follows: 3DCNN at 0.8403, PointNet at 0.8655, KPConv at 0.9064, RSConv at 0.9319, and PointNet++ at 0.9336. Notably, our newly introduced SMNet culminated in an impressive accuracy of 0.9768—a surge of 15.45% in the R2 score (Figure 3D). The details of MSE and mean absolute error (MAE) are shown in Table S1, Supporting Information. We can find that the MSE of SMNet at 0.0078 mm is merely 13% of the previous MLP's 0.0595 mm, indicating a significant decrease of more than 7.5 times.

To validate the performance of SMNet for 3D PSM devices, we implemented a virtual model of a cube composed of six 2D PSM devices, with each face featuring an independent 6×6 array of ionic actuators (Figure 3E). The simulation principle for this 3D PSM device echoes that of the 2D counterpart. Through COMSOL, we conducted simulations with 20 000 randomized control vectors and extracted the corresponding postdeformation point cloud data. Additionally, another 100 simulations were executed to serve as the test set. We compared the 100 input vectors from the test set with the model's predicted vectors. To offer an unambiguous view of the error for each actuator on every face, Figure 3F unfolds the cube, with the topmost layer representing the cube's upper face, the four intermediary layers showcasing the lateral faces, and the bottommost layer representing the cube's base. Furthermore, the predicted control vectors were fed into the simulation model to recreate the PSM deformations. The disparity between the point cloud data of these reproduced

shapes and the test set materialized as an error cloud map, as depicted in Figure 3G. Given our data's simulation origin, we possessed point cloud coordinates both pre- and postdeformation. Based on the predeformation cube coordinates, each face was segmented into 30×30 smaller squares. The average error between the reproduced data and test set data within each square was computed and color-coded on the error cloud map. As illustrated in Figure 3G, we presented the cubic error cloud map from two distinct angles, ensuring visibility of the discrepancies across all six faces. Given that the eight vertices of the ionic 3D PSM device were held fixed during simulation, the peripheral errors were minimal, with the bulk of discrepancies centralized on the squares' central regions. The average deformation magnitude for each face was around 0.4, while the peak error value was 0.03. A comparative assessment with other models was also undertaken, with R2 scores illustrated in Figure 3H. The details of MSE and MAE of each model are shown in Table S2, Supporting Information.

2.4. Model Performance of Pneumatic 3D PSM and Thermal 3D PSM Devices

In this study, we aim to propose a universally applicable, model-free technique for controlling all 3D PSM devices. To demonstrate this generality, we investigated two additional actuation mechanisms that are common in soft robotics: thermal actuation and pneumatic actuation.

First, we investigated thermal actuator arrays whose deformation principle is based on volume change, with paraffin wax serving as the primary material due to its linear and significant

thermal expansion and contraction within certain temperature bounds.^[53] In our demonstration, each thermal actuator was fashioned as a $1 \times 1 \times 1$ cube, with 152 of these units adorning the surface of a larger $6 \times 6 \times 6$ cube. Notably, the core of this assembly was a static $4 \times 4 \times 4$ passive cube unaffected by temperature-driven volume alterations. The design choice to place actuators solely on the surface stemmed from our focus on capturing the external point cloud transformations, as real-world data predominantly provides external surface point clouds. Internal actuator modifications were deemed to have limited and unclear impacts on this external point cloud. Each of the 152 surface actuators could be individually temperature-controlled, with thermal cross-talk negated by a thin insulative layer ensuring no inadvertent temperature-driven effects on adjacent units (Figure 4A).

Second, we delved into pneumatic soft actuator arrays operating on a bulking deformation principle. Similarly, 152 $1 \times 1 \times 1$ pneumatic chambers were placed on the $6 \times 6 \times 6$ cm cube's surface, each capable of independent pressure modifications. Detailed simulation setups for both mechanisms will be elaborated upon in the Section 4 (Figure 4B).

For both mechanisms, we generated 20 000 sets of point cloud data through FEA simulations as training dataset by inputting randomly generated control signal arrays, while designating 100 sets as our test dataset. After getting the pretrained model, we employed it to forecast our test dataset, resulting in 100

predictions. These predicted values were then integrated into their respective FEA models to generate point clouds of the reproduced shapes. A comparative analysis, akin to the methodology outlined in the previous section, was conducted between the predictions and test dataset, as well as input point clouds and reproduced point clouds. The findings from this comparative study are depicted in Figure 4C–F. In the case of the thermal mechanism, the fixed point is situated at the model's center, allowing for an unhindered movement of all points on the surface. Consequently, the errors of predictions and the reproduced point clouds in relation to the ground truth exhibit a uniform distribution across the surface (Figure 4C,D). Figure S1A, Supporting Information shows an example of a comparison between ground truth and reproduced point cloud. However, for the pneumatic mechanism, its fixed points align with those of the ionic mechanism, situated at the eight vertices of the cube. However, unlike the ionic mechanism where each actuator deforms relatively independently, the bulking deformation principle inherent to the pneumatic mechanism results in substantial coupling between the deformations of actuators. As a consequence, the deformation near the 12 edges is relatively limited, leading to a slight decline in prediction accuracy for the edge chambers compared to the central chambers (Figure 4E). Nevertheless, for the reproduced point clouds, given the negligible deformation along the edges, the primary deviations are predominantly centered but remain minimal (Figure 4F). Also, the

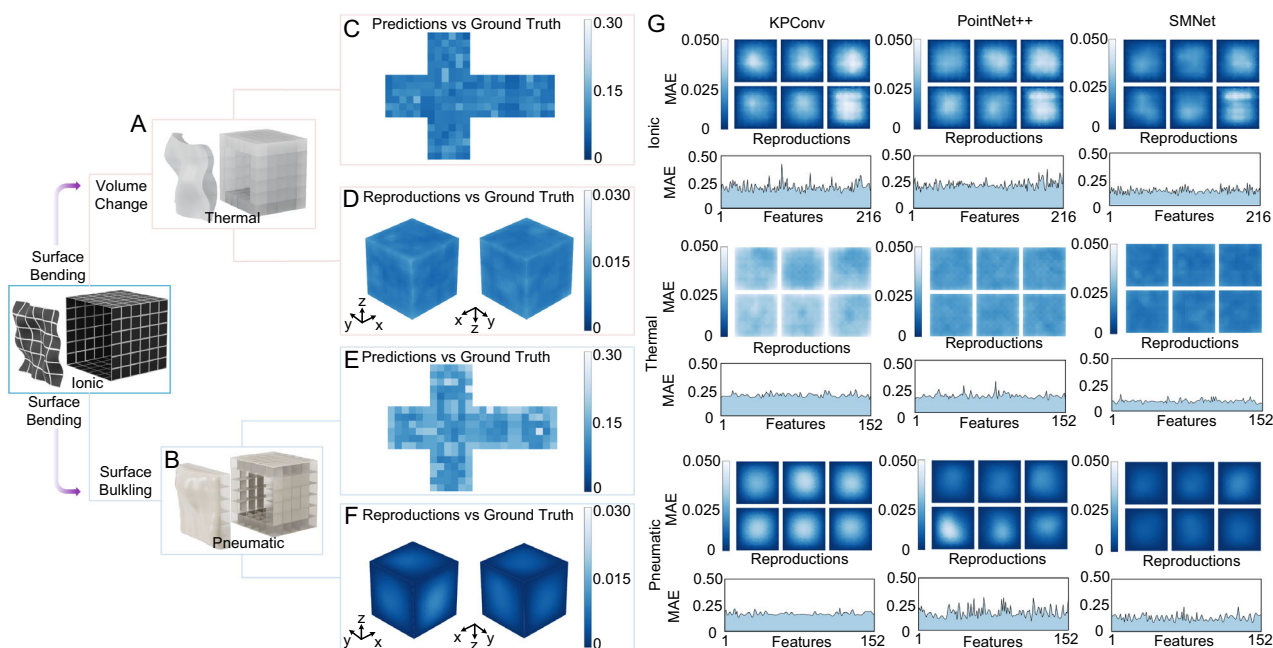


Figure 4. The model performance of thermal 3D PSM device and pneumatic 3D PSM device and the comparison between the model performance of SMNet with KPConv and PointNet++. A) Expanding from ionic 3D PSM device based on the bending principle to thermal 3D PSM device based on volume change. B) Expanding from ionic 3D PSM device based on the bending principle to pneumatic 3D PSM device based on surface buckling. C,E) The unfolded error map between model predictions and the ground-truth control input vectors of thermal 3D PSM device and pneumatic 3D PSM device, respectively. D,F) The 3D error map of thermal 3D PSM device and pneumatic 3D PSM device showcasing the point cloud of reproduced shapes with the ground-truth point cloud. There are two angles of view to show the entire 6 surfaces of the cube. G) The model performance comparison between KPConv, PointNet++, and SMNet. We laid out the error maps of the reproduced point cloud and the ground-truth point cloud into six faces, arranged in two rows. Additionally, we compared each dimension of the predicted input vector with the ground-truth input vector, and linearly displayed the error of each dimension below the point cloud error maps.

overall deformation of the pneumatic mechanism possesses lower surface complexity compared with the other two mechanisms. An example of a comparison between ground truth and reproduced point cloud is shown in Figure S1B, Supporting Information.

Given that KPConv and PointNet++ are sublayers of SMNet, we specifically compared the performance of these three models across different mechanisms. As shown in Figure 4G, we laid out the error color maps of the reproduced point cloud and the ground-truth point cloud into six faces, arranged in two rows. We divided each face of the cube into a 30×30 grid, and then calculated the average error between the reproduced point cloud and the ground-truth point cloud within each grid. Different error values were represented using different colors. Through the color map, it can be clearly seen that SMNet has a significantly lower average error compared to the other two methods. Additionally, we compared each dimension of the predicted input vector with the ground-truth input vector, and linearly displayed the error of each dimension below the point cloud error maps. Since all the figures are of the same scale, it is evident that SMNet demonstrates the best performance in terms of prediction accuracy. The specific training data (MSE, MAE, R2 score) for the three models across the three mechanisms are shown in Table S3, Supporting Information.

We also evaluated the number of trials required for training, as shown in Figure S2A, Supporting Information. For 2D PSM device, a dataset of 5000 trials proved sufficient for different models to achieve high prediction accuracy. The significant improvement in prediction accuracy was observed between 1000 and 3000 trials. For 3D PSM devices as depicted in Figure S2B, Supporting Information, which entail higher complexity, larger training sets are necessary. In the case of the thermal mechanism, where actuator coupling is relatively low, a dataset of 10 000 trials suffices to meet training requirements. However, for the Ionic mechanism, an example of high actuator coupling requiring the most inputs, a training set of 20 000 trials is needed to reach an acceptable accuracy level.

2.5. Inverse Demos for 3D PSM Devices

In the context of shape-morphing devices, the paramount capability is to mimic the morphology of target shape which is also called the inverse control. To validate the broad applicability of our SMNet for 3D PSM devices, it is imperative for the system to adapt to any physical shape found in the real world. This aspiration aligns with one of the ultimate objectives in the domains of soft robotics, biomimetic robots, and haptic devices.

To facilitate this, we manually molded clay to create a target 3D shape. Using a 3D scanner, we captured the resultant shape of the clay in the form of point cloud data. After preprocessing the data as illustrated in Figure 5A, it was input into our pretrained model. The output prediction from this model is the control vector for the 3D PSM devices. For instance, under the ionic mechanism, the output represents voltage values of each pixel (216-dimensional vector), for the thermal mechanism, it is the temperature values for each small cube (152-dimensional vector), and for the pneumatic mechanism, it is the air pressure values within each chamber (152-dimensional vector). These control

vectors are then separately input into the FEA models of these mechanisms, revealing the morphed outcomes upon specifying a target shape.

Demos 1 and 2 present two shapes of varying complexities formed by manually molding the clay. The 3D scanned images of these physical shapes and the reproduced point cloud representations from the three mechanisms are showcased in Figure 5B. To elucidate the deformation effect in the reproduced point cloud images, we calculated the displacement values of each point before and after deformation and represented this using a color gradient. Through the colored point cloud, it can be seen that all three mechanisms are capable of reproducing the main features of demos 1 and 2.

However, since manually molded clay might lack intricate detail, in demo 3, we employed Autodesk Maya software to create a 3D model, which prominently features the word “PURDUE” protruding on its six faces. The reproduced point cloud results of the three mechanisms for this design are depicted in the third column of Figure 5B. It is evident from these reproduced point clouds that the ionic and thermal mechanisms, owing to the relative independence of each actuator and minimal mechanical coupling with adjacent actuators, offer superior programmability. The shapes from all three demos were impeccably replicated by these mechanisms. Conversely, the pneumatic mechanism, characterized by significant mechanical coupling between actuators, could only reproduce a relatively simple shape (demo 1). It manifested discernible deviations from the target shape in demo 2, and demo 3 was entirely beyond its morphing capacity, resulting in a shape bearing scant resemblance to the original. The detailed procedures of demos are shown in Movie S1–S3, Supporting Information.

To quantitatively assess the fidelity of the three mechanisms in replicating the three demos, we subjected the original point clouds and their respective reproductions to a similarity analysis. In the realm of point cloud similarity metrics, three primary distances are prominently employed: chamfer distance (CD),^[54] standard deviation of distance, and Hausdorff distance (HD).^[55] The detailed explanation of these metrics is illustrated in Section S2, Supporting Information. The specific data for these three metrics are displayed in Figure 5C. Due to the pneumatic mechanism’s demo 3 failing to achieve the target shape, its error significantly exceeds that of the other cases.

We also explored the feasibility of achieving real-time control with our developed models. To this end, we compared the training duration and demonstration execution times for three distinct models, as detailed in Table S4, Supporting Information. While the training time of these models necessitates a considerable time investment, executing pretrained models requires just over one second. This response time falls below the actuation times of both ionic and thermal actuators. Since the control execution is faster than the physical actuation of the device, the model can be used for real-time control.

Furthermore, our analysis revealed that predicting outcomes for 100 sets of data in parallel only incurs an additional delay of ≈ 0.5 s compared to processing a single dataset. This suggests that if a series of target shapes can be input simultaneously, the real-time responsiveness of the control system could be significantly enhanced. Therefore, the results lay a solid foundation for implementing real-time control in 3D PSM devices, thereby expanding their practical applications in various domains.

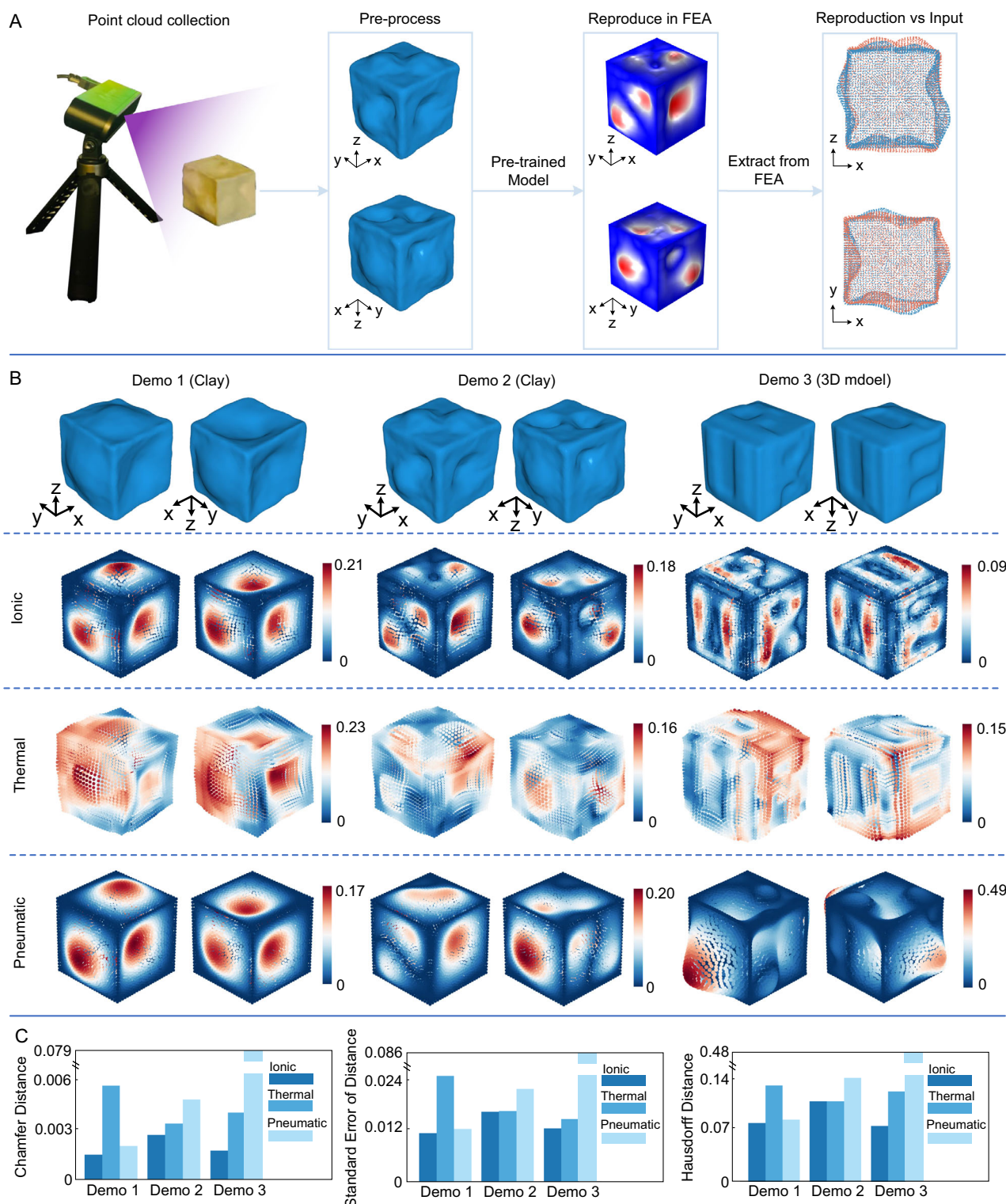


Figure 5. The demonstration of SMNet on inverse control of 3D PSM devices. A) The detailed procedures of the demonstrations. B) The reproduced point cloud of 3 different mechanisms. “Demo 1” and “demo 2” present two shapes formed by manually molding the clay. “Demo 3” is made by software with high surface complexity. C) The similarity between the reproduced point cloud with the target point cloud by using CD, standard deviation of distance, and HD. All of the data have been normalized to 1. To facilitate a better comparison among the other cases, the bars for pneumatic actuators employ a truncated axis.

3. Discussion

This study introduces a universal approach to control 3D PSM devices across various actuation principles. By leveraging point cloud to describe the deformation and deep learning techniques, we have developed a point cloud regression model, SMNet, to map desired 3D shapes to the high-dimensional input vectors that represent control inputs, providing a model-free control methodology that can be applied to shape-morphing devices with various actuation mechanisms. SMNet's ability to handle intricate geometric couplings and deformations enables superior performance over shape-morphing devices when comparing it with existing models. Additionally, in this work, we demonstrated our proposed method enables morphology mimicking of physical objects and complex virtual 3D models with high fidelity on different actuation mechanisms.

In this article, the training data were generated from FEA models because it offers an automated approach to generate large datasets that are free from noise and nonidealities such as manufacturing variations. Our approach is enabled by advancements in FEA that have significantly improved the accuracy of simulations.^[56,57] However, our proposed method would also be compatible with source data from physical devices, with the drawback of long times required to collect the training data.

Compared to traditional mathematical modeling, our method offers significant advantages. Mathematical models become increasingly challenging as the coupling between actuators increases and as the shapes become more complex, such as moving from 2D to 3D shape morphing. The development of mathematical models is time-consuming and often requires numerous assumptions that can diminish the precision of inverse control. Our proposed methodology simplifies this process by requiring only the creation of a simulation model, which is then used to generate a dataset through repeated computations with different inputs. This dataset is subsequently trained using our SMNet. For instance, in the case of the ionic 3D PSM device explored in this article, the simulation time for a single result was ≈ 5 min. We utilized a server powered by two AMD 7H12 CPUs for parallel computations, acquiring 20 000 datasets within 5 days. The training phase, when conducted using multiple H100 parallel processors, could be condensed to ≈ 3 –4 days. As such, a fully operational control model for a highly complex 3D PSM device can be developed in around 10 days, significantly streamlining the process compared to traditional methods.

In an inverse prediction task, the error includes a contribution from the accuracy of the prediction model as well as a contribution from the limitations of the actuator mechanism. For example, an actuator that can only achieve a small bending deformation will not be able to produce a shape that requires sharp features. The demos that were prepared with three different actuator mechanisms (ionic, thermal, and pneumatic) therefore have different abilities to reproduce the target shapes based on the limitations of their actuation mechanisms. The complexity of a surface can be quantified by the variance in normal vectors, as described in Section S3 and Figure S4, Supporting Information. The surface complexity of the demos and the reproduced shapes are included in Table S5, Supporting Information. The pneumatic actuation mechanism consistently achieves a surface complexity that is significantly lower than the target shape.

This emphasizes a key advantage of this generalized approach for shape-morphing control: it allows comparison of different actuation mechanisms to facilitate the selection of an appropriate actuation mechanism for the target geometry.

In this work, we use SMNet to find the relationship between the input control parameters of a 3D actuator array and the point cloud describing the deformed geometry. Modified versions of SMNet may find widespread value in predicting the deformation of other soft continuum structures, such as the deformation of biological systems during growth based on input parameters such as the location and type of cells. Future work can therefore investigate the use of SMNet with cloud point datasets that include points within the interior of a structure that can be extracted from 3D imaging techniques.

4. Experimental Section

FEA Simulation of Actuation Mechanisms: The simulation of ionic actuators is described in detail in our previous work.^[21] The architecture of the ionic actuator adopts a sandwich configuration. The outer layers are composed of a conductive electrode that swells in response to a voltage, for which we use the materials properties of polypyrrole (PPy). The middle layer is an ionic conductor and electrical insulator, for which we use the materials properties of porous poly(vinylidene fluoride) (PVDF). In our simulation, the thickness of the central PVDF layer is set at 110 microns and that of the electrode is 20 microns. Material-wise, the Young's modulus for PVDF is 2.45 GPa and for PPy it is 2 GPa. A Poisson Ratio of 0.25 was uniformly assigned to both materials. For the sake of simulation in ionic actuators, thermal expansion is conventionally used as a surrogate for its electrical expansion. Drawing from our prior experimental data,^[21] the thermal expansion coefficient for PPy is defined as 0.05, while for PVDF it is established at $1.2\text{e-}6$ based on its inherent properties. Both materials have their specific heat capacities set at $4200\text{ J kg}^{-1}\text{ K}^{-1}$. Structurally, the cubic framework is assembled from six individual square panels, each hosting a 6×6 array of discrete ionic actuators. Every standalone ionic actuator is square-shaped, with a side-to-gap ratio of 10:1. This simulation was conducted in COMSOL, opting for a tetrahedral mesh with the default coarser size setting. The applied voltage on a pixel was assigned as a boundary condition with half of the requisite voltage applied to the outer PPy regions and half to the inner sections. For instance, if a pixel's target voltage is -0.6 V, the distribution would be -0.3 V at the outer surface and 0.3 V at the inner surface. Mechanical boundary conditions consisted of fixed constraints at the eight vertices of the cube.

As to the thermal actuator array, we chose Abaqus as the simulation software. In this case, we aim to simulate the shape morphing caused by temperature-induced volume changes. The structure consists of a cubic geometry with each face consisting of 6×6 addressable pixels. Given our focus on surface deformations, only the pixels on the cube's surface are actuated, while the interior of the cube is made of an undeformable and temperature-insensitive material. For the thermal actuators on the surface layer, we employed materials exhibiting linear thermal expansion properties, exemplified by paraffin wax actuators. The thermal expansion coefficient of the actuating material was chosen as 0.4 to normalize the deformation range to input temperatures in the bound of ± 1 °C, and we assume it has high thermal conductivity that allows the temperature in this cube to be uniform. To ensure that the temperature of each actuating pixel is independent, we added a very thin layer of material with super-low thermal conductivity between the pixels. Those intersecting sheets have a thickness of 0.5% of the thermal actuator length. To simulate the thermal expansion of this actuator array with unique temperatures assigned to each actuating pixel, a "coupled temperature-displacement" step was created in Abaqus to analyze the steady-state response, with automatic incrementation, maximum number of increments set to 100, and a minimum increment size of $1\text{e-}5$, with other setting being defaulted. The unique

temperatures informing the thermally isolated expansion of each actuator were addressed by iterating over each actuator's surfaces normal to the Z direction in the script and applying the chosen randomized temperatures as "temperature boundary conditions" in Abaqus. As for mechanical boundary conditions, we apply an "encastre boundary condition" at the very central nonactuating point of the $6 \times 6 \times 6$ grid, thus constraining the location without affecting the deformation due to the outer actuators. A very coarse mesh size of a quarter of the actuator length was chosen to reduce the time of a single simulation since the training requires 20 000 trials, and the element type is chosen as "C3D8T."

Within the pneumatic actuator array, we implemented a hollow structure in the core. Chambers are only established on the surfaces of the cube (6×6 in each surface). Owing to the shared chambers at the edges and corners, the total number of independently controlled chambers is 152. We used the materials properties of Sylgard 184, a common silicone elastomer used in soft robotics. Sylgard 184 has a typical Young's modulus of 2 MPa and a Poisson's ratio of 0.48. As this simulation was also executed in COMSOL, the mesh choice remained a tetrahedral mesh with the default coarser size. The pneumatic inputs were normalized to a range between -1 and 1 to prevent gradient explosions during subsequent machine learning training phases. Additionally, constraints were affixed at the eight vertices of the cube.

The examples of training datasets for three different mechanisms collected from simulation results are shown in Figure S3, Supporting Information.

Principle of SMNet: In the KPConv layer, we have the point coordinates $\{x_i\}$ drawn from a point cloud $\mathcal{P} \in \mathbb{R}^{N \times 3}$. Correspondingly, we construct a feature vector f_i for each point, initially filled with ones, denoted by the set $\{\mathcal{F} \in \mathbb{R}^{N \times 1} | f_i = 1\}$. We introduce a kernel within a predefined radius $r \in \mathbb{R}$, and define the neighborhood N of a point x within this radius as $\mathcal{N} = \{x_i \in \mathcal{P} | \|x_i - x\| \leq r\}$, where x_i and x denote the coordinates used to calculate Euclidean distances. The convolution of feature f by the kernel g centered at x is thus formulated as

$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}} g(x_i - x) f_i \quad (1)$$

The kernel function g operates on the relative positions of neighboring points, which are computed as $y_i = x_i - x$. Its domain is the sphere $\mathcal{B}_r^3 = \{y_i \in \mathbb{R}^3 | \|y_i\| \leq r\}$. We represent kernel points as $\{\tilde{x}_k\}$ with the constraint $\{\tilde{x}_k | k < K\} \subset \mathcal{B}_r^3$, where K signifies the total number of kernel points. The associated weight matrices that project features from the input dimension, which is 4 (considering a supplementary 1D feature vector of one's appended to the point coordinates), to an output dimension D_{out} are expressed as $\{W_k | k < K\} \subset \mathbb{R}^{4 \times D_{out}}$. Accordingly, the kernel function g can be formulated as^[51]

$$g(y_i) = \sum_{k < K} h(y_i, \tilde{x}_k) W_k \quad (2)$$

Here, h signifies the correlation between kernel points $\tilde{x}_k$ and the relative position y_i . A linear correlation function is applied as

$$h(y_i, \tilde{x}_k) = \max\left(0, 1 - \frac{\|y_i - \tilde{x}_k\|}{\sigma}\right) \quad (3)$$

where σ denotes the influence distance of the kernel points. In this study, K is selected to be 15, and σ is chosen as 1.5. Typically, the larger the kernel size, the more local geometric information the convolutional kernel can capture. However, if the geometric details of the point cloud are not highly intricate, it is unnecessary to select a very large kernel size. A common choice ranges between 8 and 16. After fine-tuning, we found that a kernel size of 15 is a parameter that balances training accuracy with a reasonable training time. σ is related to the kernel radius r , where r equals to σ multiply the grid size. This parameter varies across layers. We set the initial grid size to 0.02 and σ as 1.5, so the radius of the first layer is $0.02 \times 1.5 = 0.03$. The subsequent grid sizes are 0.04, 0.08, 0.16, and 0.32, leading to radii of 0.06, 0.12, 0.24, and 0.48 in the following layers.

As mentioned above, the base architecture of KPConv is U-Net, which contains three encoding layers and two decoding layers, and there is a KPConv in each layer. In each encoding layer, multiscale precomputed data would go through the down convolution while in each decoding layer, upsample precomputed data would take the upconvolution. The final output of KPConv I is a 6D feature $\mathcal{F}_{out} \in \mathbb{R}^{N \times 6}$.

In the layers processing PointNet++, the input is a point set $\mathcal{P}_f \in \mathbb{R}^{N \times 9}$ combined the newly generated features $\mathcal{F}_{out} \in \mathbb{R}^{N \times 6}$ with the original point cloud $\mathcal{P} \in \mathbb{R}^{N \times 3}$ to a new point sets $\mathcal{P}_f \in \mathbb{R}^{N \times 9}$. As aforementioned, the PointNet++ layer contains the sampling, grouping, and PointNet process. In sampling process, the input point set is $\mathcal{P}_f \in \mathbb{R}^{N \times 9}$. By utilizing the iterative farthest point sampling (FPS), a subset of points with number of $N_s, \mathcal{P}_n = \{X_n | n < N_s\} \subset \mathcal{P}_f$, can be chosen. In this subset, the point x_{ni} is the most distant point (in metric distance) from the set $\mathcal{P}_{nj} = \{X_{nj} | j < i\} \subset \mathcal{P}_n$ with regard to the rest points.^[58] The pseudocode is shown below **Algorithm 1**

In the grouping stage, we group the original point $\mathcal{P}_f \in \mathbb{R}^{N \times 9}$ set and the coordinates the centroids of the subset $\mathcal{C} \in \mathbb{R}^{N_s \times 3}$. The output of grouping could be $\mathcal{G} \in \mathbb{R}^{N' \times M \times 9}$, where M is the number of points in the neighborhood of centroid points.^[50] Given that our point clouds contain $\approx 5,000$ – $7,000$ points, we set N_s to 2,048 in the first layer, and then reduce it progressively to 1,024, 512, and 256 in subsequent layers. The radii of grouping are 0.2, 0.4, 0.8, and 1.2 and M is 64, 32, 16, and 16.

In the PointNet layer, the coordinates of points in a local region are first translated into $y_i = x_i - x$ where x is the coordinates of the central points.

$$f(y_1, y_2, \dots, y_n) = \text{MLP}(\max(\text{MLP}(y_i))) \quad (4)$$

where MLP refers to MLP networks.

After going through multiple aforementioned steps, the final output is the feature vector $\mathcal{F}'_{out} = (f_1, f_2, \dots, f_n)$ with dimensional of 1024 ($\mathcal{F}'_{out} \in \mathbb{R}^{N \times 1024}$). We utilize average pooling for these output feature vectors.

$$\bar{f}_i = \frac{1}{|N(p_i)|} \sum_{j \in N(x_i)} f_j \quad (5)$$

where $N(x_i)$ is the number of neighborhood points of point x_i .

The final step of SMNet is the fully connected layer to map the abstracted features of point cloud to the ground-truth features. The loss function of the regression procedure is chosen as MSE:

Algorithm 1. Farthest point sampling (FPS).

procedure FPS ($\mathcal{P}_f, N_s$)

$\mathcal{P}_n \leftarrow \emptyset$

Select a random point x_i from $\mathcal{P}_f$ and add it to $\mathcal{P}_n$

for all $x \in \mathcal{P}_f$ **do**

$[d] \leftarrow \|x_i - x\|$

end for

while $|\mathcal{P}_n| < N$ **do**

$x_{farthest} \leftarrow \arg \max([d]) (x \in \mathcal{P}_f \setminus \mathcal{P}_n)$

Add $x_{farthest}$ to $\mathcal{P}_n$

for all $x \in \mathcal{P}_f \setminus \mathcal{P}_n$ **do**

$[d] \leftarrow \min([d], \|x_{farthest} - x\|)$

end for

end while

return $\mathcal{P}_n$

end procedure = 0

$$L(\mathbf{v}) = \frac{1}{N} \sum_{i=1}^N (\mathbf{v}_i - \hat{\mathbf{v}}_i)^2 \quad (6)$$

where $\mathbf{v}_i$ is the output features from SMNet and $\hat{\mathbf{v}}_i$ is the ground-truth features.

Training Setting-Ups: In our study, the training architecture was constructed using PyTorch. The model was trained on an RTX 3090 GPU. Due to memory constraints associated with the RTX 3090, we set the batch size to 8. For ionic 2D low-profile PSM, the epochs of training are 200 while for all 3D PSM cases, the epochs of training are 600. The optimization was performed using Stochastic Gradient Descent (SGD) with a learning rate of 0.1 and a momentum of 0.9.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

This work was supported by the Purdue startup funding to Alex Chortos and by NSF award 2301509.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Jue Wang: Conceptualization (lead); Data curation (lead); Investigation (lead); Methodology (lead); Software (lead); Validation (lead); Visualization (lead); Writing—original draft (lead); Writing—review and editing (lead). **Dhirondaatto Sarkar:** Software (supporting); Validation (supporting). **Jiaqi Suo:** Validation (supporting); Visualization (supporting). **Alex Chortos:** Conceptualization (supporting); Funding acquisition (lead); Project administration (lead); Supervision (lead); Writing—review and editing (equal).

Data Availability Statement

The data that support the findings of this study are openly available in Zenodo at <https://zenodo.org/records/10558468>, reference number 10558468.

Keywords

3D programmable shape morphing, inverse control, machine learning, point cloud

Received: July 8, 2024
Revised: August 26, 2024
Published online:

- [1] B. Mazzolai, A. Mondini, F. Tramacere, G. Riccomi, A. Sadeghi, G. Giordano, E. Del Dottore, M. Scaccia, M. Zampato, S. Carminati, *Adv. Intell. Syst.* **2019**, *1*, 1900041.
- [2] J. Pikul, S. Li, H. Bai, R. Hanlon, I. Cohen, R. F. Shepherd, *Science* **2017**, *358*, 210.

- [3] S. R. Klemmer, B. Hartmann, L. Takayama, in *Proc. 6th Conf. Design. Interact. Syst.* ACM Press, New York, USA, **2006**, pp. 140–149.
- [4] X. Yu, Z. Xie, Y. Yu, J. Lee, A. Vazquez-Guardado, H. Luan, J. Ruban, X. Ning, A. Akhtar, D. Li, *Nature* **2019**, *575*, 473.
- [5] C. Peng, *Ph.D. Thesis*, Massachusetts Institute of Technology, Boston, USA **2020**.
- [6] X. Chen, S. Han, W. Wu, Z. Wu, Y. Yuan, J. Wu, C. Liu, *Small* **2022**, *18*, 2106824.
- [7] A. Kirillova, L. Ionov, *J. Mater. Chem. B* **2019**, *7*, 1597.
- [8] J. M. Viola, C. M. Porter, A. Gupta, M. Alibekova, L. S. Prah, A. J. Hughes, *Adv. Mater.* **2020**, *32*, 2002195.
- [9] M. J. Ford, C. P. Ambulo, T. A. Kent, E. J. Markvicka, C. Pan, J. Malen, T. H. Ware, C. Majidi, *Proc. Natl. Acad. Sci. U.S.A.* **2019**, *116*, 21438.
- [10] A. Kotikian, R. L. Truby, J. W. Boley, T. J. White, J. A. Lewis, *Adv. Mater.* **2018**, *30*, 1706164.
- [11] M. Coelho, H. Ishii, P. Maes, in *CHI'08 Extend. Abstracts Human Factors Comput. Syst.*, **2008**, pp. 3429–3434.
- [12] C. Yu, Z. Duan, P. Yuan, Y. Li, Y. Su, X. Zhang, Y. Pan, L. L. Dai, R. G. Nuzzo, Y. Huang, H. Jiang, J. Rogers, *Adv. Mater.* **2013**, *25*, 1541.
- [13] A. Nojoomi, H. Arslan, K. Lee, K. Yum, *Nat. Commun.* **2018**, *9*, 1.
- [14] Y. Mao, Z. Ding, C. Yuan, S. Ai, M. Isakov, J. Wu, T. Wang, M. L. Dunn, H. J. Qi, *Sci. Rep.* **2016**, *6*, 1.
- [15] J. Wu, C. Yuan, Z. Ding, M. Isakov, Y. Mao, T. Wang, M. L. Dunn, H. J. Qi, *Sci. Rep.* **2016**, *6*, 1.
- [16] A. A. Stanley, K. Hata, A. M. Okamura, in *2016 IEEE Int. Conf. Robot. Automat. (ICRA)*, IEEE Piscataway, USA **2016**, pp. 2718–2724.
- [17] K. Liu, F. Hacker, C. Daraio, *Sci. Robot.* **2021**, *6*, eabf5116.
- [18] X. Ni, H. Luan, J.-T. Kim, S. I. Rogge, Y. Bai, J. W. Kwak, S. Liu, D. S. Yang, S. Li, S. Li, Z. Li, Y. Zhang, C. Wu, X. Ni, J. Rogers, *Nat. Commun.* **2022**, *13*, 5576.
- [19] A. M. Rauf, J. S. Bernardo, S. Follmer, in *2023 IEEE Int. Conf. Robot. Automat. (ICRA)*, IEEE, Piscataway, USA **2023**, pp. 2591–2597.
- [20] Y. Bai, H. Wang, Y. Xue, Y. Pan, J.-T. Kim, X. Ni, T.-L. Liu, Y. Yang, M. Han, Y. Huang, J. Rogers, X. Ni, *Nature* **2022**, *609*, 701.
- [21] J. Wang, M. Sotzing, M. Lee, A. Chortos, *Sci. Adv.* **2023**, *9*, eadg8019.
- [22] K. Hirota, M. Hirose, in *Proc. Fifth Int. Conf. Artificial Reality Tele-Existence*, ICAT Committee, Tokyo, Japan **1995**, pp. 185–192.
- [23] H. Iwata, H. Yano, F. Nakaizumi, R. Kawamura, in *Proc. 28th Annu. Conf. Comput. Graph. Interact. Tech.*, ACM Press, New York, USA **2001**, pp. 469–476.
- [24] D. Leithinger, H. Ishii, in *Proc. Fourth Int. Conf. Tangible Embedded Embodied Interact.*, ACM Press, New York, USA **2010**, pp. 221–222.
- [25] S. Follmer, D. Leithinger, A. Olwal, A. Hogge, H. Ishii, in *Uist*, **2013**, Vol. 13, pp. 2501–988.
- [26] Y. Yang, Y. Wu, C. Li, X. Yang, W. Chen, *Adv. Intell. Syst.* **2020**, *2*, 1900077.
- [27] J. Wang, J. Suo, A. Chortos, *IEEE Robot. Automat. Lett.* **2021**, *7*, 549.
- [28] E. Hajiesmaili, N. M. Larson, J. A. Lewis, D. R. Clarke, *Sci. Adv.* **2022**, *8*, eabn9198.
- [29] A. Chortos, J. Mao, J. Mueller, E. Hajiesmaili, J. A. Lewis, D. R. Clarke, *Adv. Funct. Mater.* **2021**, *31*, 2010643.
- [30] A. Kotikian, J. M. Morales, A. Lu, J. Mueller, Z. S. Davidson, J. W. Boley, J. A. Lewis, *Adv. Mater.* **2021**, *33*, 2101814.
- [31] H. Jiang, Z. Wang, X. Liu, X. Chen, Y. Jin, X. You, X. Chen, in *2017 IEEE Int. Conf. Robot. Automat. (ICRA)*, ACM Press, New York, USA **2017**, pp. 6127–6133.
- [32] R. F. Reinhardt, J. J. Steil, *Proc. Technol.* **2016**, *26*, 12.
- [33] Y. Xu, X. Tong, U. Stilla, *Automat. Construct.* **2021**, *126*, 103675.
- [34] Y. Zeng, Y. Hu, S. Liu, J. Ye, Y. Han, X. Li, N. Sun, *IEEE Robot. Automat. Lett.* **2018**, *3*, 3434.
- [35] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, M. Bennamoun, *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *43*, 4338.

- [36] H. Duan, P. Wang, Y. Huang, G. Xu, W. Wei, X. Shen, *Front. Neurobot.* **2021**, 15, 658280.
- [37] F. Pomerleau, F. Colas, R. Siegwart, *Found. Trends Robot.* **2015**, 4, 1.
- [38] J. Zhang, X. Zhao, Z. Chen, Z. Lu, *IEEE Access* **2019**, 7, 179118.
- [39] Y. Xie, J. Tian, X. X. Zhu, *IEEE Geosci. Remote Sens. Mag.* **2020**, 8, 38.
- [40] E. Grilli, F. Menna, F. Remondino, *Int. Arch. Photogram. Remote Sens. Spat. Inf. Sci.* **2017**, 42, 339.
- [41] H. Zhang, C. Wang, S. Tian, B. Lu, L. Zhang, X. Ning, X. Bai, *Displays* **2023**, 79, 102456.
- [42] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, J. A. Levine, A. Sharf, C. T. Silva, in *35th Annu. Conf. Eur. Assoc. Comput. Graph., Eurographics 2014*, The Eurographics Association, Geneva, Switzerland **2014**.
- [43] Z. Ma, S. Liu, *Adv. Eng. Inf.* **2018**, 37, 163.
- [44] S. Oehmcke, L. Li, J. C. Revenga, T. Nord-Larsen, K. Trepekli, F. Gieseke, C. Igel, in *Proc. 30th Int. Conf. Adv. Geograph. Inf. Syst.*, ACM Press, New York, USA **2022**, pp. 1–4.
- [45] K. Lv, M. Yu, Y. Pu, X. Jiang, G. Huang, X. Li, in *2023 IEEE Int. Conf. Robot. Automat. (ICRA)*, IEEE, Piscataway, USA **2023**, pp. 7119–7125.
- [46] X. Chen, G. Wang, C. Zhang, T.-K. Kim, X. Ji, *IEEE Access* **2018**, 6, 43425.
- [47] L. Ge, H. Liang, J. Yuan, D. Thalmann, in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, IEEE, Piscataway, USA **2017**, pp. 1991–2000.
- [48] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, B. Chen, in *Advances in Neural Information Processing Systems*, Neural Information Processing Systems Foundation (NeurIPS), San Diego, CA, USA **2018**, p. 31.
- [49] C. R. Qi, H. Su, K. Mo, L. J. Guibas, in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, IEEE, Piscataway, USA **2017**, pp. 652–660.
- [50] C. R. Qi, L. Yi, H. Su, L. J. Guibas, in *Advances in Neural Information Processing Systems*, Neural Information Processing Systems Foundation (NeurIPS), San Diego, CA, USA **2017**, p. 30.
- [51] H. Thomas, C. R. Qi, J.-E. Deschaut, B. Marcotegui, F. Goulette, L. J. Guibas, in *Proc. IEEE/CVF Int. Conf. Comput. Vision*, IEEE, Piscataway, USA **2019**, pp. 6411–6420. Publisher: IEEE, Publisher Location: Piscataway, USA.
- [52] Y. Liu, B. Fan, S. Xiang, C. Pan, in *Proc. IEEE/CVF Conf. Comput. Vision pattern Recogn.*, IEEE, Piscataway, USA **2019**, pp. 8895–8904. Publisher: IEEE, Publisher Location: Piscataway, USA.
- [53] A. Mann, C. M. Bürgel, P. Groche, in *Actuators* **2018**, 7, 81.
- [54] T. Wu, L. Pan, J. Zhang, T. Wang, Z. Liu, D. Lin, in *Advances in Neural Information Processing Systems*, Neural Information Processing Systems Foundation (NeurIPS), San Diego, CA, USA **2021**, vol. 34, 29088.
- [55] F. Mémoli, G. Sapiro, in *Proc. 2004 Eurographics/ACM SIGGRAPH Symp. Geometry Process.*, ACM Press, New York, USA **2004**, pp. 32–40.
- [56] M. S. Xavier, A. J. Fleming, Y. K. Yong, *Adv. Intell. Syst.* **2021**, 3, 2000187.
- [57] P. Boyraz, G. Runge, A. Raatz, *Actuators* **2018**, 7, 48.
- [58] Y. Eldar, M. Lindenbaum, M. Porat, Y. Y. Zeevi, *IEEE Trans. Image Process.* **1997**, 6, 1305.