Design of Fully Controllable and Continuous Programmable Surface Based on Machine Learning

Jue Wang, *Student Member, IEEE*, Jiaqi Suo, and Alex Chortos, *Member, IEEE*

***Abstract*—Programmable surfaces (PSs) consist of a 2D array of actuators that can deform in the third dimension, providing the ability to create continuous 3D profiles. Discrete PSs can be realized using an array of independent solid linear actuators. Continuous PSs consist of actuators that are mechanically coupled, providing deformation states that are more similar to real surfaces with reduced complexity of the control electronics. However, continuous PSs have been limited in size by the lack of the control systems required to take into account the complex internal coupling between actuators in the array. In this work, we computationally explore the deformation of a fully continuous PS with 81 independent actuation pixels based on ionic bending actuator. We establish a control strategy using machine learning (ML) regression models. Both forward and inverse control are achieved based on the training datasets which are derived from the finite element analysis (FEA) data of our PS. The prediction of surface deformation achieved by forward control with accuracy under 1% is 15000 times faster than FEM. And the real-time inverse control of continuous PSs that is to reproduce any arbitrary pre-defined surfaces, which possess high practical value for tactile display or human-machine interactive devices, is first proposed in the letter.**

***Index Terms*—Soft robot applications, AI-based methods, machine learning for robot control.**

I. Introduction

THE original concept of programmable surface (PS) first occurred in Ivan Sutherland's article which was depicted as 'Ultimate Display'. [1] With further research in this field, the programmable surface is defined as a tactile display interface that creates and updates physical shapes, with the ability to interact with objects via the variation of the physical shape of the surface itself. It has wide prospects for applications in human-computer interaction devices and graphic display.

Shape morphing describes the process of transitioning between 3D shapes driven by the design of the materials or structure of the system. Shape morphing between two specific shapes can be accomplished by locally tailoring the response of a structure to an external stimulus. For example, locally programming the orientation of ferromagnetic domains achieves tailorable local curvatures in response to a magnetic field. Encoding of magnetic domains has been achieved using applied magnetic fields at the nozzle of a 3D printer [2] and direct laser writing [3] in a sheet of magnetic soft composite. To tailor the response of materials to electric fields, the local mechanical properties can be programmed using 3D printed stiff fibers [4] or uniaxial alignment of polymer chains using light exposure [5]. These examples illustrate the versatility of shape morphing strategies based on local programming of materials properties. However, these structures, once programmed, can only distort into one shape. Localized on-demand actuation is necessary to allow morphing into an infinite number of shapes.

For PS, the individual actuation of each pixel is critical to achieving the "programmable" characteristic. They can be divided into two categories: discrete and continuous surfaces. Discrete surfaces consist of a matrix of linear actuators that control the height of each pixel independently. This concept was first proposed by Hirota *et al.* in 1993. They use a 4×4 matrix linear actuator to form random surface [6]. After that, many prototypes, such as FEELEX [7], Relief [8] and inFORM [9] have been developed with higher accuracy (more actuators) and more complex applications. The actuator mechanism also varies from traditional motor to pneumatic linear actuator [10], [11], SMA [12] and DEAs [13]. The control algorithms for these systems are simple because of the independent control of each pixel. However, the matrix of linear actuators takes up a lot of space and the individual control of each actuator requires a large control system. Hence, discrete PSs are difficult to incorporate into wearable devices or other micro-devices. In addition, the precision of such programmable surfaces depends on the number of actuators.

Continuous PSs consist of arrays of coupled actuators. While discrete PSs consist of linear actuators that are perpendicular to the surface, continuous PSs can be created using linear or bending actuators that are oriented in the plane of the surface. Coelho *et al.* presented the most basic continuous PS with four independent SMA actuators which made up a square. [14] A soft robotic surface driven by pneumatic bending actuators which was able to achieve several basic surfaces was proposed by Chen *et al.* [15] Liu *et al.* developed a continuous PS driven by grid of bending actuators driven by heat-responsive LCEs with stretchable heating coils. [16] Even though the aforementioned continuous PSs possess higher accuracy theoretically, current examples can achieve a limited number of patterns due to the small number of actuators and the undiscovered complex

Manuscript received July 11, 2021; accepted November 10, 2021. Date of publication November 22, 2021; date of current version December 3, 2021. This letter was recommended for publication by Associate Editor C. Cao and Editor K.-J. Cho upon evaluation of the reviewers' comments. This work was supported by startup funds from Purdue University. *(Corresponding author: Jue Wang; Alex Chortos.)*

Jue Wang and Alex Chortos are with the College of Engineering, Purdue University 500 Central Drive, West Lafayette, IN 47907 USA (e-mail: wang5056@purdue.edu; achortos@purdue.edu).

Jiaqi Suo is with the Gensler Baltimore, 1 East Pratt Street Suite 202, Baltimore, MD 21202 USA (e-mail: suojiaqi0710@gmail.com).

Digital Object Identifier 10.1109/LRA.2021.3129542

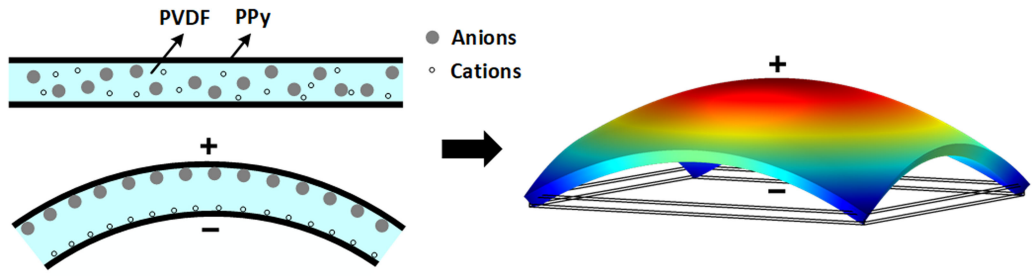

Fig. 1. The deformation principle of square PPy actuator.

internal coupling. As the number of pixels in a continuous PS increases, the control algorithms will become a limiting factor.

Unlike discrete PSs, in which the height value of each pixel can be independently controlled, the analytical algorithms for controlling continuous PSs are difficult to derive due to implicit couplings between pixels [17]. Thus, inspired by I. M. Van Merbeek's work [18], which used ML algorithms to reveal the internal relationship of passive matrix sensors, we use ML to analyze the complicated internal coupling of a matrix of bending actuators.

Hence, in this letter, given that discrete PSs have limited accuracy and shape-morphing robots and current continuous programmable surfaces cannot be fully controlled, we propose an approach based on ML to control a continuous PS with 81 pixels. The whole mechanism only has four fixed vertices as the boundary conditions. Thus, how the deformation of one pixel will affect the other pixels around it cannot be calculated by an analytical model, and the finite element analysis (FEA) simulation is time-consuming for online control. Hence, several ML regression models are used to predict the deformation both forwardly and inversely where the training datasets are derived from the FEA simulation. The performance illustrates that both forward and inverse control can be achieved in real-time with high precision.

The key novel contributions are summarized as follows:

- This is the first proposed continuous PS that can be fully controlled, since the complex internal coupling between each actuator in matrix is determined through ML algorithms.
- The forward control demonstration shows that the ML-based control system is able to predict the deformation of complicated surfaces online with high accuracy. In contrast, similar studies can only form some simple pre-defined surfaces.
- The inverse control demonstrations show that any arbitrary surface can be reproduced, which suggests potential value in tactile displays and human-machine interactive devices.

II. Mechanism and Method

A. Actuator Mechanism

We base our continuous PS simulations on the deformation mechanism of ionic electro-active polymer actuators (IEAPAs) due to their compatibility with bending actuator geometries. IEAPAs convert electrical inputs into mechanical deformation. The electric field causes a movement of ions, which induces movement of the solvent and consequent swelling or contraction in the membrane or electrodes. This ionic polarization provides large bending displacement at low voltages. [19], [20] However, low force and the requirement of encapsulation or protective layer in the open air conditions [21] limit its application fields.

In this work, we develop our FEM model based on the actuation mechanism of polypyrrole (PPy) actuators which can deform to both sides. The principle of PPy actuator is shown in Fig. 1. The function of the middle layer, Polyvinylidene Fluoride (PVDF), is to insulate both sides of the film while absorbing and storing electrolyte. When there is a potential difference between the two sides of PPy, the electrochemical reaction between the PPy and electrolyte inside the PVDF on the anode and cathode causes the migration of anions and cations, which can generate volume change on both sides.

As shown in Fig. 2, the mechanism we simulated possess laminated structure with 3 layers. But when fabricating the practical prototype, it can be beneficial to include a gold layer to increase the conductivity and enable electrochemical deposition of PPy. [22] To simplify the control system and the structure, passive matrix addressing can be applied for actuator matrix. In the middle of Fig. 2, the input voltage can be applied directly on the edge of top and bottom PPy layer. Since the ionic actuators require a closed-loop circuit, the line input should have the common ground wire. In this way, each active area can be driven independently with the potential difference of two inputs. For example, to drive a particular pixel, the corresponding cross of top and bottom PPy strips should be activated. By using passive matrix addressing, independent control of each pixel can be realized. In addition, there is no need to connect the wires to each active area, which will have a great influence on the control accuracy since the actuation force of the PPy actuator is small. However, in a passive matrix, crosstalk must be considered, which will be discussed in the Section IV In this case, there are 9 horizontal and 9 longitudinal strips of PPy on either side of the ionic dielectric. These crossed layers form 9×9 active pixels. In our simulation work, we treat the passive addressing as the direct addressing shown on the right of Fig. 2. Meanwhile, in order to ensure the uniqueness of its solution, its four vertices are fixed as boundary conditions in the simulation.

B. Simulation of Deformation

FEA is the most direct way to predict surface deformation. However, FEA can only predict the deformation when the input

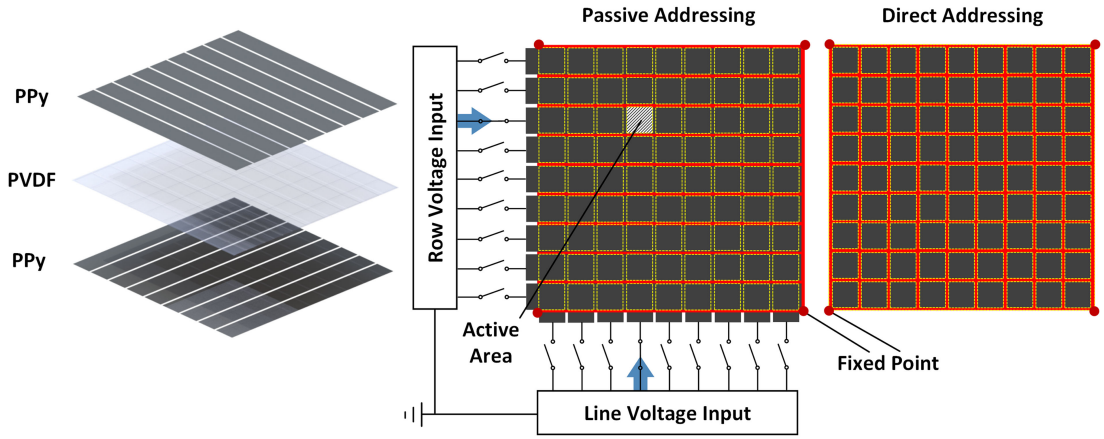

Fig. 2. The laminated structure of PPy actuator and structure of continuous programmable surface with 9×9 pixels.

voltages are determined. Inverse control, which uses the desired surface deformation as the input to calculate the necessary driving voltages, cannot be achieved. In addition, the large computational time for FEA makes it difficult to realize real-time control. Current real-time FEAs are achieved by adding simplifying assumptions and sacrificing the number of nodes, which limit the accuracy of the simulations and the complexity of the mechanisms that can be simulated. [23], [24] Thus, in this work, we use FEA results as the training data for the ML regression model, and then execute the model to control the programmable surface in real-time.

The simulation of IEAPAs has previously been accomplished using a two-step process. The first step is to simulate the ionic concentration caused by potential difference. Such ionic transport process can be calculated by Nernst-Planck equation. This can be achieved in COMSOL Multiphysics using the Electrostatics and the Transport of Diluted Species physics in a 1D model. It is necessary to enable the Potential Coupling and the Space Charge Density Coupling multiphysics. From this simulation, we can obtain the ionic concentration throughout the device perpendicular to the applied field. Then, in a 2D or 3D model, the strain can be calculated through the Solid Mechanics physics based on the relationship between force and ionic concentration. The drawback of this approach is its difficulty in achieving the simulation directly in a 3D model. Thus, we follow the work of Madden *et al.*, [25] which introduces a simplified approach. The diffusive elastic metal model shows that in a certain range of deformation, the strain of PPy actuator generated via ionic transport is proportional to the applied voltage: $\epsilon_V = \beta \Delta V$, where ϵ_V is the strain indirectly caused by voltage change ΔV, and β is the electrical expansion coefficient. [25] Thermal expansion has has the same mathematical form as ion-induced strain: $\epsilon_T = \alpha \Delta T$, where ϵ_T is the strain generated by temperature change ΔT, and α is the thermal expansion coefficient. Hence, if we only focus on the deformation of the structure rather than the ionic transport process, as shown in Fig. 3, the electrical field can be substituted by the thermal field, where thermal strain is equivalent to electrical strain and thermal expansion coefficient is equivalent to electrical expansion coefficient.

To set up the thermal simulation, we implemented the Heat Transfer in Solids and Solid Mechanics physics as well as Thermal Expansion and Temperature Coupling multiphysics. The

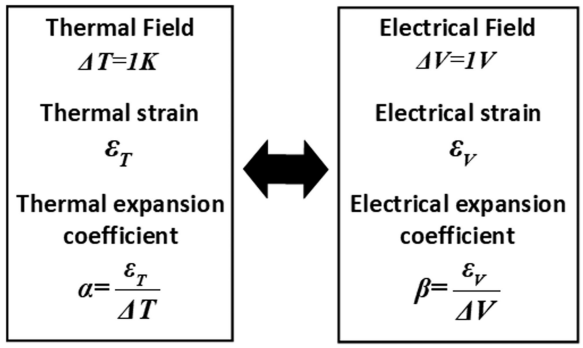

Fig. 3. The equivalent substitution principle between electrical expansion and thermal expansion.

TABLE I
TABLE OF SIMULATION PARAMETERS

Parameter	Value	Unit	
k_1	1.15	$W/(m \cdot K)$	
k_2	0.21	$W/(m \cdot K)$	
ρ_1	1400	kg/m^3	
ρ_2	1780	kg/m^3	
C_{p1}	1200	$J/(kg \cdot K)$	
C_{p2}	1400	$J/(kg \cdot K)$	
E_1	80e6	Pa	
E_2	2.7e9	Pa	
ν_1	0.4	1	
ν_2	0.4	1	
α_1	0.060	$1/K$	
α_2	1.3e-4	$1/K$	

3D model was built in Solidworks and then imported to COMSOL. The mesh we used in this work was physics-controlled and normal size. We chose quasistatic study and the result of displacement field was our ultimate target. Here, the parameters of aforementioned thermal simulation is shown in Table. I.

Here, k_1 and k_2 are the thermal conductivity of PPy and PVDF material respectively. ρ_1 and ρ_2 are the density of PPy and PVDF. C_{p1} and C_{p2} are the heat capacity at a constant pressure of PPy and PVDF. E_1 and E_2 are the Young's modulus of PPy and PVDF. ν_1 and ν_2 are the Poisson's ratio PPy and PVDF. α_1 and α_2 are the secant coefficient of thermal expansion of PPy and PVDF. [26] Here, the thermal expansion coefficient for PPy is not the real value. It is derived from the diffusive elastic metal

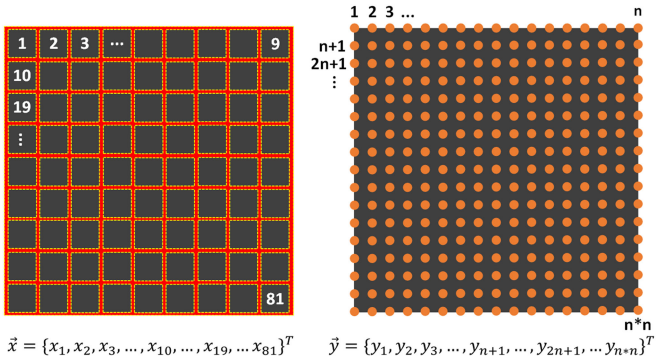

Fig. 4. The definition of training datasets including voltage array and z-displacement array.

model based on experimental data. [25], [27] While the other parameters for PPy and PVDF are all experimentally measured values of the materials.

III. Machine Learning Control

A. Generating Datasets

In this work, we use FEM simulation results to generate the datasets for the ML regression model. The dataset should include two groups of arrays. One is the voltage array which contains the voltage of each pixel. The other is the z-displacement array of surface which consists of the value of z-displacement of each selected node. As shown in Fig. 4 , these two arrays arrange the data in the same order (from left to right & from top to bottom). The dimension of voltage array equals 81, and the dimension of z-displacement array depends on the density of nodes we select $(n \times n)$. Since the dimension of both arrays are larger than one, we should implement multi-output regression model to solve it. We evaluate the use of k-nearest neighbors (KNN), random forest (RF) and multilayer perceptions (MLP) regression models to train our dataset.

In these two arrays, the voltage array is randomly generated and the z-displacement array can be obtained by FEM simulation. First, we define an array which is $(-2, -1.9, \ldots, 1.9, 2)$ in MATLAB. Then we randomly pick a value in this array for each pixel as its input voltage and combine them into the voltage array. Thus, the average distribution of this voltage array is uniform since each value in it is picked randomly in a certain range. The next step is to implement the voltage arrays in COMSOL to derive the z-displacement arrays through FEM simulation. Here, to increase the simulation accuracy, we divide each value in the voltage array by two and then apply them to top layer and bottom layer respectively. (For example, if the voltage value is 1.6 V, we apply 0.8 V on the top layer and $-0.8\,V$ on the bottom layer) Through COMSOL Multiphysics with MATLAB, the entire simulation process can be executed in a loop. The distribution of z-displacement arrays is not standard uniform since there are some boundary conditions in the set-up of the simulation, which caused the variance to be non-uniform, but the mean of such arrays perform uniformly, which is close to zero.

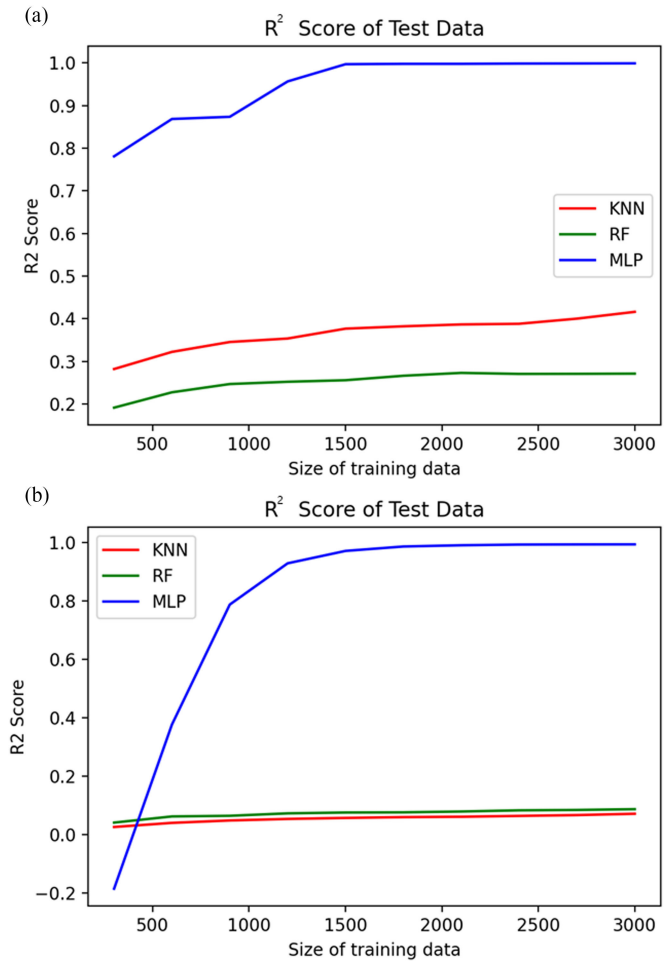

Fig. 5. The R^2 score of KNN, RF and MLP regression model. (a) is the forward control. (b) is the inverse control.

B. Training Datasets

In our model, we have 81 pixels, and the value of each pixel can be from $-2\,V$ to $2\,V$. If we collect value from the array mentioned above, an exhaustive search of all possible training datasets would have required 41^{81} trials. Making an exhaustive dataset requires too much computational capacity, especially for the FEA simulation. Thus, we choose 3000 trials for each regression model. This reduction of the necessary size of the datasets is one of the significant advantages of ML algorithms. Meanwhile, we use scikit-learn in Python to implement these ML regression models. The details are shown below:

- *KNN:* For both forward and inverse control, the number of neighbors is 5 and the weight is 'distance.'
- *RF:* For both forward and inverse control, we just choose default settings.
- *MLP:* The number of hidden layers is 5. The neurons for forward control are 420, 350, 240, 160 and 90, respectively. While the neurons for inverse control are 90, 160, 240, 350, and 420, respectively.

The forward control, which utilizes input voltage to obtain a prediction of a surface, can be achieved through either FEA simulations or a combination of FEA simulations and ML algorithms. The advantage of ML algorithms is that its

TABLE II
THE MEAN TIME OF TRAINING AND EXECUTING MODELS

	KNN	RF	MLP
Training Time Complexity	$\mathcal{O}(knp)$	$\mathcal{O}(n^2pN_{tree})$	$\sim\mathcal{O}(n^2(pN_{l1}+N_{l1}N_{l2}+...))$
Executing Time Complexity	$\mathcal{O}(knp)$	$\mathcal{O}(pN_{tree})$	$\mathcal{O}(pN_{l1}+N_{l1}N_{l2}+...)$
Forward Training	0.0059s	122.01s	2869.22s
Inverse Training	0.0060s	240.95s	2647.71s
Forward Executing	0.0050s	0.00011s	0.0060s
Inverse Executing	0.0059s	0.00010s	0.0039s

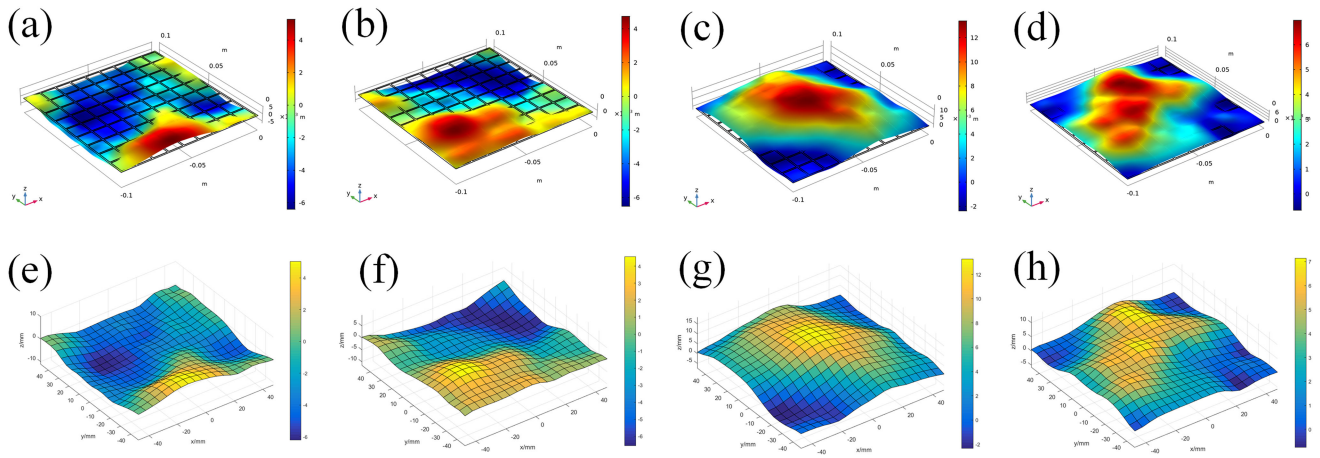

Fig. 6. The application of forward control: predicting the surface deformation based on random input voltage arrays. (a)-(d) are the control group of surface simulated by COMSOL. (e)-(f) are the surface generated by the predicted z-displacement arrays.

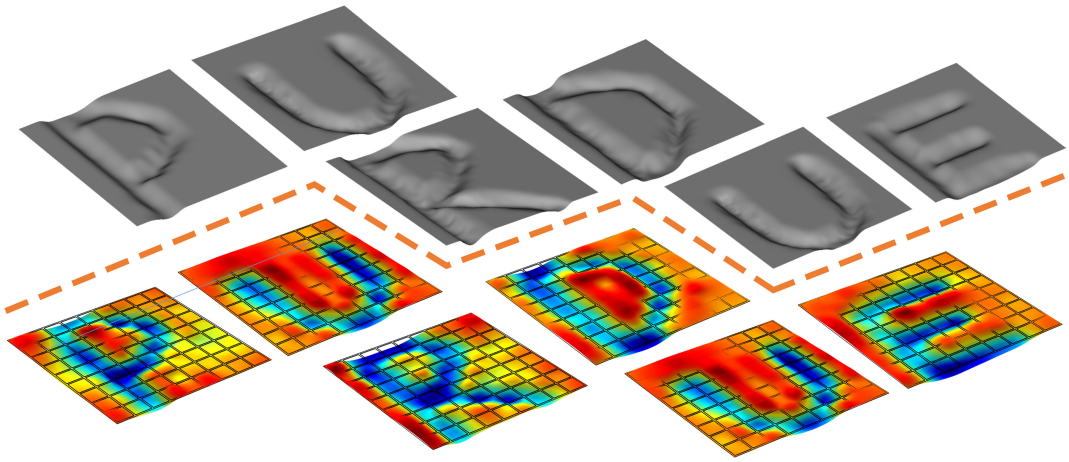

Fig. 7. The application of inverse control: Display the letters based on the surface created manually. The top figures are the letter surfaces generated in Maya. The bottom figures are the letters displayed by proposed PS.

computational efficiency (executing ML models, not training ML models) can realize online control. ML algorithms cannot completely replace FEM, which is necessary to train the ML algorithms. Nevertheless, the inverse control, which uses a desired surface to calculate voltage inputs, can only be realized by ML algorithms. Generally, inverse control possesses higher application value because it allows the formation of any arbitrary surface that is desired. Thus, in this work, aiming to achieve forward control, we firstly define the voltage arrays as the input data, while the z-displacement arrays are treated as output data. Then we turn z-displacement arrays to input data and voltage arrays to output data to train the inverse control. The details of the training results will be shown in the next section.

IV. RESULTS

A. Model Performance

To evaluate the performance of each regression model, we use the R^2 score since it is a unitless method to reflect the relationship between the prediction data and the test data. In comparison, mean square error (MSE) and root mean square error (RMSE) give absolute values. The closer the R^2 score

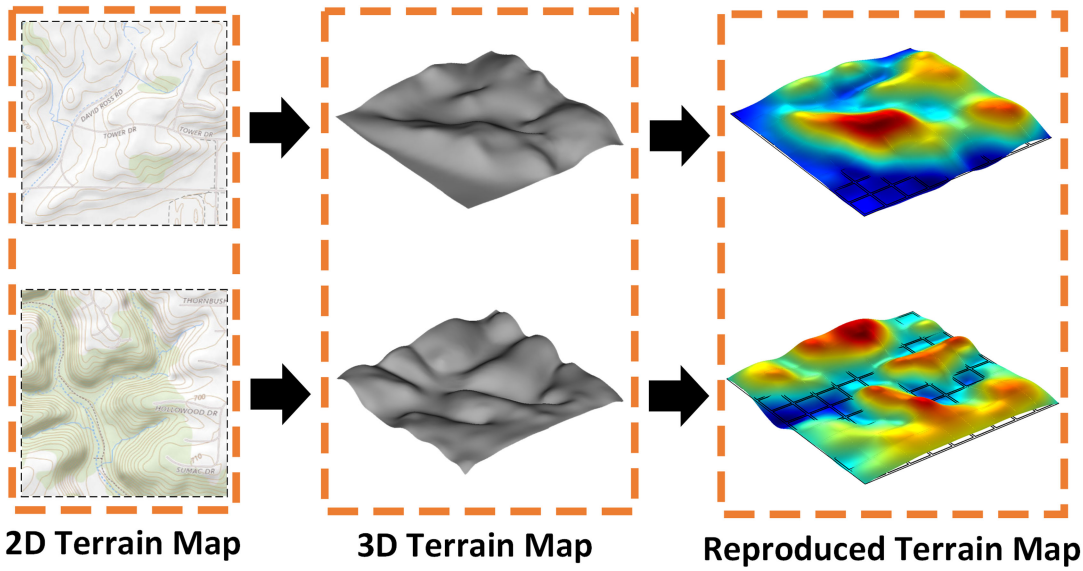

Fig. 8. The application of inverse control: Display terrain maps based on the captured 2D maps.

is to 1, the more the prediction data matches the test data, indicating a good model. As shown in Fig. 5, the R^2 score of the KNN and RF regression model illustrates that such models are not able to achieve prediction since their R^2 score are all smaller than 0.5. In comparison, the MLP regression model performs well, exhibiting an R^2 score ¿0.9 with large datasets. It indicates that for such high-dimension multi-output regression problems, traditional machine learning models cannot work with the small-size datasets (compared with the exhaustive search), but the deep learning (DL) models with artificial neural network (ANN) are more suitable for these cases. As depicted in Fig. 5, the performance of MLP improves with the increase of the size of training data. For forward control, small size of training data still possesses a quite tolerable R^2 score, and the R^2 score exceeds 0.9 when the size is around 1000. On the other hand, for inverse control, the MLP models become more acceptable (> 0.9) only when the size of training data increases to about 1200 and the small size of training data cannot achieve prediction.

As for the computational cost of the ML algorithms, the time complexity and the actual running time of each algorithm is shown in Table. II. Here, the n is the number of samples, the p is the number of features, the k is number of neighbors in KNN, the N_{tree} is the number of trees in RF and the N_{li} is the number of neurons at layer i in a neural network. (The time complexity of training MLP is an estimate and may not be very accurate.) Based on the time complexity, it can be found that when array density increases, the number of features p and number of neurons at each layer N_{li} increases, so that the time complexity of training each model increases. Besides, the Table. 2 also shows that the theoretical and experimental computational time match each other well. For example, the time complexity of training and executing KNN model is the same, while the running time of these two are also the same. Besides, the time complexity of executing these three models are on the same order of magnitude because they do not contain n^2 terms. Meanwhile, the execution time of these three models are on the same magnitude and are much smaller than the training time of RF and MLP which contain the n^2 terms in their time complexity. The most important information in this table is that the mean execution time of these models are all smaller than 0.01 s, which means the online control can be at least 100 Hz.

B. Demonstrations of Forward Control

As mentioned above, the applications of forward control are limited since we can only use it to predict the surface deformation rather than to produce a desired surface. Here, to show the accuracy of our forward control, we derive the z-displacement arrays from the 300 groups of input voltage arrays and then reproduce the z-displacement arrays to surface using MATLAB. Meanwhile, we also use these input voltage arrays to simulate the surface in COMSOL as the control group.

The MSE of these two groups of surfaces (z-displacement arrays) is 0.012 mm. Compared with the overall size of the mechanism in the x-y direction (100 $mm \times 100\,mm$), mean z-displacement (from $-11.15\,mm$ to 11.58 mm) and minimum and maximum z-displacement ($-21.12\,mm$ and $-21.90\,mm$), the error of forward control is about 0.1% of the mean z-displacement. As shown by the R^2 data in Fig. 5, the high accuracy of our simulations is enabled by using sufficiently large training datasets. Fig. 6 shows 4 examples of 300 groups of test data. Fig. 6(a)-(d) are the control group simulated by COMSOL while Fig. 6(e)-(h) represents the surface predicted by the ML model. Through the comparison of figures, we can also intuitively find the accuracy of the ML model.

C. Demonstrations of Inverse Control

Most continuous PSs only use arched surfaces or saddle surfaces as demonstrations. [14]–[16] However, our larger array can achieve much more sophisticated surface deformations. Hence,

to show the programmable capability of our simulated PS, we design two different demonstration scenarios shown below:

1) *Forming letters on surface:* In this demonstration, we use the inverse control to reproduce the letters generated by a third 3D model software. First, we use Maya to create letter surfaces manually. Then we export the 3D model as an STL document to Meshlab. Second, we use Meshlab to transfer the STL model to cloud point with x-y-z data. Third, we import the cloud data to MATLAB to process them to be the ordered z-displacement arrays and then input arrays to inverse control ML model to derive the voltage arrays for our PS. Finally, we implement the voltage arrays into the COMSOL to show the real performance of our PS. Here, we create 'P,' 'U,' 'R,' 'D' and 'E' letters to form a word 'PURDUE' shown in Fig. 7. The top figures are the 3D model designed in Maya and the bottom figures are the surfaces formed by derived voltage arrays.

2) *Forming terrain maps on surface:* In this demonstration, the simulated PS is used to display the 3D terrain maps based on surface input. As shown in Fig. 8, first, we capture two 2D terrain maps from a website, which are located in the north and northeast of Purdue University respectively (Campus of West Lafayette). Based on these 2D terrain maps, we recreate the 3D terrain in Maya. After that, we use the same method which is introduced in the first inverse application to generate the voltage arrays of these terrain maps. The reproduced terrain maps based on the trained voltage arrays are shown in the right of Fig. 8.

As for the precision, the MSE of z-displacement arrays of 'P,' 'U,' 'R,' 'D,' and 'E' are $0.32\,mm$, $0.22\,mm$, $0.75\,mm$, $0.27\,mm$ and $0.18\,mm$, respectively. While the MSE of the two terrain maps are $1.21\,mm$ and $1.67\,mm$, respectively. Compared with the forward control, the error slightly increases, but as shown in Fig. 7&8, it is still acceptable for 3D display applications. The reason is that even though our PS possesses much more independent actuating units than other studies, it still only has the 9×9 matrix, which means the arbitrary surfaces created manually cannot be completely reproduced. However, based on the limited number of actuators, the proposed control system can still restore the general shape of the input surface.

V. Conclusion

In this work, a fully controllable and continuous PS with 81 independent actuating pixels based on ionic actuators was simulated. By using the simulation data of PS as training datasets, the MLP regression model could control the array in the forward and inverse direction in real-time with high precision. The performance of forward control showed the excellent prediction abilities of the proposed PS, where the MSE between predict surface and control group is only $0.012\,mm$ (less than 1% of mean z-displacement). Meanwhile, the inverse control applications illustrated that the PS is able to reproduce any man-made surfaces with acceptable error, which has great potential to be applied as tactile displays and human-machine interactive devices.

Currently, all of the training datasets are generated through FEA simulation. The fabrication of the physical array of actuators is still in progress, so one of the future works is to validate the simulation data as well as the training model in a real system. Besides, the density of pixel actuator and the dimension of the z-displacement array limit the accuracy of inverse control since some fine structure in arbitrarily generated surfaces cannot be fully reproduced. Increasing the number of pixels and the dimension of the z-displacement array seems to be a promising future step, but the volume of control system and the time complexity of model training will also increase. (The number of features is positively correlated with the time complexity of training model) Thus, in future work, we need to find a balance between these two aspects. In addition, if the passive addressing method is used for practical prototype, crosstalk should be considered, which means that the voltage value of a pixel can be influenced by its adjacent pixels' inputs. Progressive scan is a potential solution to the crosstalk problem. For example, the first row input can be set to $0\,V$ while the other rows are left floating, then each line input is set to the pre-defined voltage of the first row pixels. At this moment, only first row pixels can be actuated at their desired voltage. The rest of pixels cannot be driven since one input of them is at the floating status (The ionic actuator can only be driven by closed-loop circuit, since potential difference does not exist at the floating status, which cannot realize the ionic immigration inside the actuator). In the next moment, only the second row input is connected to be $0\,V$ and the line input are set to be the value of each pixels at the second row. So on and so forth, each pixel can be controlled individually, but not precisely simultaneously. As long as the scan frequency is larger than the time response of the actuator, passive addressing should perform similarly to direct addressing, where the crosstalk issue can be partially solved.

References

[1] I. Sutherland, "The ultimate display," in *Proc. IFIP Congress*, 1965, pp. 506–508.

[2] Y. Kim, H. Yuk, R. Zhao, S. A. Chester, and X. Zhao, "Printing ferromagnetic domains for untethered fast-transforming soft materials," *Nature*, vol. 558, no. 7709, pp. 274–279, 2018.

[3] H. Deng, K. Sattari, Y. Xie, P. Liao, Z. Yan, and J. Lin, "Laser reprogramming magnetic anisotropy in soft composites for reconfigurable 3D shaping," *Nat. Commun.*, vol. 11, no. 1, pp. 1–10, 2020.

[4] E. Hajiesmaili, E. Khare, A. Chortos, J. Lewis, and D. R. Clarke, "Voltage-controlled morphing of dielectric elastomer circular sheets into conical surfaces," *Extreme Mechanics Lett.*, vol. 30, 2019, Art. no.100504.

[5] Z. S. Davidson *et al.*, "Monolithic shape-programmable dielectric liquid crystal elastomer actuators," *Sci. Adv.*, vol. 5, no. 11, 2019, Art. no. eaay0 855.

[6] K. Hirota and M. Hirose, "Surface display: Concept and implementation approaches," in *Proc. 5th Int. Conf. Artif. Reality Tele-Existence*, 1995, pp. 185–192.

[7] H. Iwata, H. Yano, F. Nakaizumi, and R. Kawamura, "Project FEELEX: Adding haptic surface to graphics," in *Proc. 28th Annu. Conf. Comput. Graph. interactive Techn.*, 2001, pp. 469–476.

[8] D. Leithinger and H. Ishii, "Relief: A scalable actuated shape display," in *Proc. 4th Int. Conf. Tangible*, embedded, and embodied interaction, 2010, pp. 221–222.

[9] S. Follmer, D. Leithinger, A. Olwal, A. Hogge, and H. Ishii, "inFORM: Dynamic physical affordances and constraints through shape and object actuation," *UIST*, vol. 13, no. 10.1145, pp. 2501 988–2502032, 2013.

[10] H. Zhu and W. J. Book, "Practical structure design and control for digital clay," in *Proc. Int. Mech. Eng. Congr. Expo.*, 2004, pp. 1051–1058.

[11] Z. Deng, M. Stommel, and W. Xu, "A novel soft machine table for manipulation of delicate objects inspired by caterpillar locomotion," *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 3, pp. 1702–1710, Jun. 2016.

[12] I. Poupyrev, T. Nashida, S. Maruyama, J. Rekimoto, and Y. Yamaji, "Lumen: Interactive visual and shape display for calm computing," in *Proc. ACM SIGGRAPH Emerg. Technol.*, SIGGRAPH'04. New York, NY, USA: Association for Computing Machinery, 2004, Art. no. 17.

[13] T. Wang, J. Zhang, J. Hong, and M. Y. Wang, "Dielectric elastomer actuators for soft wave-handling systems," *Soft Robot.*, vol. 4, no. 1, pp. 61–69, 2017.

[14] M. Coelho, H. Ishii, and P. Maes, "Surflex: A programmable surface for the design of tangible interfaces," in *Proc. CHI'08 extended Abstr. Hum. Factors Comput. Syst.*, 2008, pp. 3429–3434.

[15] L. Chen, C. Yang, H. Wang, D. T. Branson, J. S. Dai, and R. Kang, "Design and modeling of a soft robotic surface with hyperelastic material," *Mechanism Mach. Theory*, vol. 130, pp. 109–122, 2018.

[16] K. Liu, F. Hacker, and C. Daraio, "Robotic surfaces with reversible, spatiotemporal control for shape morphing and object manipulation," *Sci. Robot.*, vol. 6, no. 53, 2021, Art. no. eabf5116.

[17] W. M. van Rees, E. Vouga, and L. Mahadevan, "Growth patterns for shape-shifting elastic bilayers," in *Proc. Nat. Acad. Sci.*, 2017, pp. 11 597–11602.

[18] I. Van Meerbeek, C. De Sa, and R. Shepherd, "Soft optoelectronic sensory foams with proprioception," *Sci. Robot.*, vol. 3, no. 24, 2018, Art. no. eaau2489.

[19] M. Annabestani and M. Fardmanesh, "Ionic electro active polymer-based soft actuators and their applications in microfluidic micropumps, microvalves, and micromixers: A review," *arXiv:1904.07149*, Apr. 2019.

[20] W. Hong, A. Almomani, Y. Chen, R. Jamshidi, and R. Montazami, "Soft ionic electroactive polymer actuators with tunable non-linear angular deformation," *Materials*, vol. 10, no. 6, pp. 1996–1944, 2017.

[21] E. N. Gama Melo, O. F. Aviles Sanchez, and D. Amaya Hurtado, "Anthropomorphic robotic hands: A review," *Ingeniería y desarrollo*, vol. 32, no. 2, pp. 279–313, 2014.

[22] M. Tyagi, G. M. Spinks, and E. W. Jager, "3D printing microactuators for soft microrobots," *Soft Robot.*, vol. 8, no. 1, pp. 19–27, 2021.

[23] C. Duriez, "Control of elastic soft robots based on real-time finite element method," in *Proc. IEEE Int. Conf. Robot. Automat*., 2013, pp. 3982–3987.

[24] F. Largilliere, V. Verona, E. Coevoet, M. Sanz-Lopez, J. Dequidt, and C. Duriez, "Real-time control of soft-robots using asynchronous finite element modeling," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 2550–2555.

[25] J. Madden, "Polypyrrole actuators: Properties and initial applications," in *Electroactive Polymers for Robotic Applications*. Berlin, Germany: Springer, 2007, pp. 121–152.

[26] B. Lunn, J. Unsworth, N. Booth, and P. Innis, "Determination of the thermal conductivity of polypyrrole over the temperature range 280-335 K," *J. Mater. Sci.*, vol. 28, no. 18, pp. 5092–5098, 1993.

[27] A. D. Price and H. E. Naguib, "A unified multiphysics finite element model of the polypyrrole trilayer actuation mechanism," *J. Intell. Mater. Syst. structures*, vol. 24, no. 5, pp. 548–558, 2013.