

Computational co-design of structure and feedback controller for locomoting soft robots

Yuki Sato, Changyoung Yuhn, Hiroki Kobayashi, Atsushi Kawamoto, Tsuyoshi Nomura*

Toyota Central R&D Labs., Inc., 1-4-14, Koraku, Bunkyo-ku, Tokyo, 1120004, Japan

Abstract

Soft robots have gained significant attention due to their flexibility and safety, particularly in human-centric applications. The co-design of structure and controller in soft robotics has presented a longstanding challenge owing to the complexity of the dynamics involved. Despite some pioneering work dealing with the co-design of soft robot structures and actuation, design freedom has been limited by stochastic design search approaches. This study proposes the simultaneous optimization of structure and controller for soft robots in locomotion tasks, integrating topology optimization-based structural design with neural network-based feedback controller design. Here, the feedback controller receives information about the surrounding terrain and outputs actuation signals that induce the expansion and contraction of the material. We formulate the simultaneous optimization problem under uncertainty in terrains and construct an optimization algorithm that utilizes automatic differentiation within topology optimization and neural networks. We present numerical experiments to demonstrate the validity and effectiveness of our proposed method.

Keywords: soft robots, design optimization, topology optimization, neural networks, material point method

1. Introduction

Soft robotics has attracted a growing interest, attributed to its inherent flexibility, versatility, and safety, particularly in human-centric applications. It has wide applications, including exploration tasks, object manipulators, medical usages, and wearable devices [1–3]. For designing functional soft robots, the co-design of both structure and controller has presented a persistent challenge due to the complex dynamics, including the large deformation and interaction with their surrounding environment. Cheney et al. [4] performed a pioneering study for co-designing the structure and actuation of soft robots for locomotion tasks based on the compositional pattern-producing networks (CPPN). Van Diepen and Shea [5] presented the co-designing of structures and actuation of soft robots where the structures and actuation were optimized by simulated annealing. Bhatia et al. [6] proposed a co-design method of soft robots using the CPPN for structural designs and reinforcement learning for the controller designs. They applied it for various tasks, including locomotion tasks and object manipulation tasks. Despite pioneering efforts exploring the co-design for soft robots, the degrees of design freedom for structures were relatively limited, primarily due to the reliance on stochastic design exploration methods.

As a structural optimization method with high degrees of design freedom, topology optimization [7] has gained much attention over decades. The major challenges to applying it for the structural design of soft robots are handling large deformations and contacts in forward analysis, usually performed by the finite element method (FEM) [8, 9]. While topology optimization has been originally studied in designing stiff structures [10, 11], there have been extensive studies that considered material and geometrical non-linearity

*Corresponding author.

Email address: nomu2@mosk.tytlabs.co.jp (Tsuyoshi Nomura)

for large deformations in the literature [12–15]. Several studies also handled contacts of materials with themselves or other materials in addition to large deformations [16, 17].

Material point methods (MPMs) [18–21] are potential alternatives for performing forward analysis involving large deformations and contacts of materials with other materials or boundaries. Compared with FEMs, MPMs can easily handle large deformations and contacts owing to their hybrid Lagrangian–Eulerian approach. MPMs have been incorporated in topology optimization [22] for elastic materials with large static deformations. In more recent studies, we have introduced MPMs in topology optimization for locomoting soft robots [23]. Furthermore, we have proposed *4D topology optimization* [24], which simultaneously optimizes the structure, actuator layout, and actuation pulse sequences of a soft robot by representing them as multi-index density variables in four dimensions (i.e., space and time). The key technique of these studies is the incorporation of MPMs with automatic differentiation, which assists in computing gradients involving forward analysis even for highly complex dynamics including large deformations and contacts. The experimental demonstration has also been reported, which enhanced the effectiveness of this approach [25]. Previous efforts, however, have focused on optimizing actuation signals for a predetermined environment rather than creating a controller system that is adaptable to environmental changes. Hence, the optimized soft robots were operated by feedforward controllers and may be less robust under uncertainties in environments.

To address the environmental uncertainties, we focus on the machine learning (ML)-based controllers for soft robots given that ML techniques have been widely used for soft robot control in the literature [26–28]. Indeed, previous works for the co-design of soft robots mentioned above successfully dealt with the soft robot control based on the ML-based techniques [4–6]. However, the effectiveness of such ML techniques for topology optimization is currently controversial, especially for direct design models without using physical properties [29] although there are several previous efforts [30–32]. Therefore, it is important to consider how to integrate ML-based techniques into the well-established gradient-based topology optimization for structural designs.

In this study, we propose simultaneous optimization of both structure and feedback controller for soft robots involving locomotion tasks, integrating gradient based-topology optimization for structural designs with neural network-based controller designs. Meanwhile, we use neural networks for feedback controller designs [27, 33], specifically a multilayer perceptron [34]. The neural network-based controller generates actuation signals based on terrain features surrounding the soft robots, inducing the expansion and contraction of the soft bodies. We also optimize the actuator layout based on our previous method [24]. Figure 1 illustrates the concept of our proposed method.

The contributions of our current work are as follows:

- We formulate the co-design problem of a structure and controller of a soft robot for locomotion tasks. We set the design space of structure and actuator layouts based on the topology optimization approach while we model the feedback controller using neural networks.
- We construct an optimization algorithm for co-designing soft robots that move over various terrains. To account for the uncertainty of the terrain, our algorithm prepares random terrain datasets and trains neural networks simultaneously with the structure and actuator layout.
- We demonstrate the effectiveness of our proposed method by providing forward analysis results using test terrain datasets that are not used in the training (optimization) process.

The remainder of this paper is organized as follows. In Section 2, we explain the governing equations for forward problems, which include the material and actuation models. We also briefly discuss the discretization of the governing equation using MPMs. In Section 3, we formulate the optimization problem for co-designing soft robots. We also explain the dataset preparation of varying terrains and construct the optimization algorithm of the structures, actuator layouts, and feedback controllers of soft robots. In Section 4, we provide the numerical experiments to demonstrate the validity and the effectiveness of our proposed method. Finally, we conclude this study in Section 5.

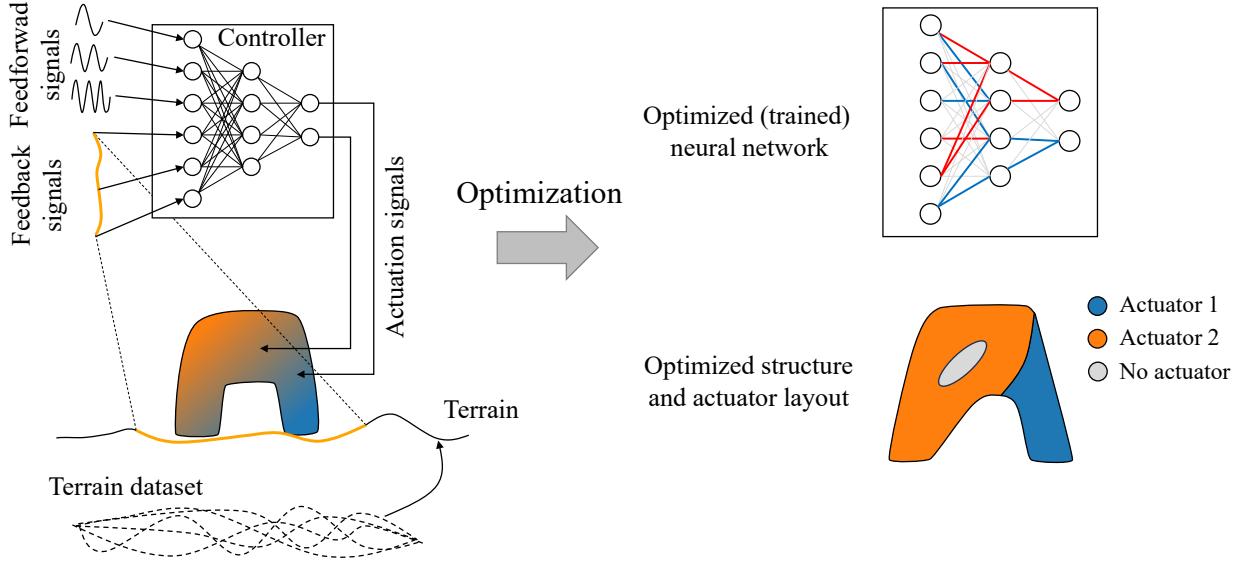


Figure 1: Conceptual diagram of our proposed method.

2. Forward problem

2.1. Governing equations

Let $\Omega(t)$ denote a material domain at time t . The conservation laws of mass and momentum in the domain $\Omega(t)$ are given as

$$\frac{D\rho(t, \mathbf{x})}{Dt} + \rho(t, \mathbf{x})\nabla \cdot \mathbf{v}(t, \mathbf{x}) = 0, \quad (1)$$

$$\rho(t, \mathbf{x})\frac{D\mathbf{v}(t, \mathbf{x})}{Dt} = \nabla \cdot \boldsymbol{\sigma}(t, \mathbf{x}) + \mathbf{f}(t, \mathbf{x}), \quad (2)$$

where \mathbf{x} is the spatial coordinate, ρ is the material density, \mathbf{v} is the velocity, $\boldsymbol{\sigma}$ is the Cauchy stress tensor, and \mathbf{f} is the external force vector. The operator D/Dt denotes the material derivative. Hereinafter, we will omit the arguments (t, \mathbf{x}) if apparent from the context. Assuming the hyperelastic material governed by the neo-Hookean material constitution, we derive the Cauchy stress tensor via the potential energy Ψ and the deformation gradient tensor \mathbf{F} , as follows:

$$\boldsymbol{\sigma} = \frac{1}{J} \frac{\partial \Psi}{\partial \mathbf{F}} \mathbf{F}^\top + \eta \left(\nabla \mathbf{v} + \nabla \mathbf{v}^\top - \frac{2}{3} (\nabla \cdot \mathbf{v}) \mathbf{I} \right), \quad (3)$$

$$\Psi(\mathbf{F}) := \frac{\mu}{2} \left(\text{tr}(\mathbf{F}^\top \mathbf{F}) - d \right) - \mu \log(J) + \frac{\lambda}{2} \log^2(J), \quad (4)$$

where $J := \det(\mathbf{F})$, λ and μ are the Lamé constants, d is the spatial dimension, η is a damping parameter and \mathbf{I} is the identity matrix with the size of $d \times d$. The second term of the Cauchy stress is the viscosity term for numerical damping which is added to avoid numerical instabilities of high-frequency oscillations. For the external force, we applied the gravity and the self-actuating force as

$$\mathbf{f}(t, \mathbf{x}) = -\nabla \cdot \boldsymbol{\sigma}^{\text{act}}(t, \mathbf{x}) + \rho(t, \mathbf{x})\mathbf{g}, \quad (5)$$

$$\boldsymbol{\sigma}^{\text{act}}(t, \mathbf{x}) := -a(t, \mathbf{x})\mathbf{F}(t, \mathbf{x})\mathbf{S}\mathbf{F}(t, \mathbf{x})^\top, \quad (6)$$

where $a(t, \mathbf{x})$ is the actuation signal, \mathbf{g} is the gravity acceleration and \mathbf{S} is the second Piola–Kirchhoff stress tensor prescribing the deformation caused by the actuation. In the present study, we set $\mathbf{S} := \mathbf{I}$. Since we

consider the locomoting soft robots, we have to consider boundary conditions that describe the interaction of the soft robots with the boundaries of the environment, e.g., ground and wall. In the present study, we consider the no-slip ground that imposes

$$\mathbf{v}(t, \mathbf{x}) = \mathbf{0} \text{ for } \mathbf{x} \text{ on the ground.} \quad (7)$$

2.2. Discretization

To solve the governing equations in Eqs. (1) and (2) with the boundary condition in Eq. (7), we used the moving least square material point method (MLS-MPM) [21], which is a variant of material point methods (MPMs) [18–20], a Lagrangian-Eulerian hybrid method. MPMs discretize the material domain into particles and take into account the couples of particles using a background grid. More precisely, the numerical simulation process of MPMs is the iteration of the following steps:

(1) Particle to grid

Transfer the mass and momentum of particles to nodes of a background grid.

(2) Update velocities

Calculate the velocities on each node of the grid by transferred mass and momentum. Update the velocity on the grid considering the external forces and the boundary conditions.

(3) Grid to particle

Transfer the updated velocity on nodes of the grid to each particle and update positions of each particle.

The boundary condition in Eq. (7) is imposed in the second step by replacing the grid velocities on the boundary, as follows:

$$\text{For } m \in \mathcal{B}, \quad \mathbf{v}_m \leftarrow \begin{cases} \mathbf{0} & \text{if } \mathbf{v}_m \cdot \mathbf{n}_m < 0 \\ \mathbf{v}_m & \text{otherwise,} \end{cases} \quad (8)$$

where \mathcal{B} is the index set of grid nodes on the boundary, \mathbf{v}_m is the velocity on the m -th node of the grid and \mathbf{n}_m is the normal vector at the m -th node pointing outward to the boundary. This boundary condition corresponds to the no-slip boundary condition and can be easily extended more general boundary condition considering the Coulomb friction [21]. See the reference [21] for more detailed implementation.

3. Optimization problem

Let us introduce a fixed design domain Ω_D that includes the material domain at an initial time t_0 , i.e., $\Omega(t_0) \subset \Omega_D$. We now describe how to optimize the structure, actuator layout, and controller of a soft body.

3.1. Topology optimization

We formulate the structural optimization problem of the soft body based on the density-based topology optimization approach [35, 36]. Let $\gamma_i \in [0, 1]$ be the fictitious material density for the i -th particle where $\gamma_i = 1$ for solid particles, $\gamma_i = 0$ for void particles and $0 < \gamma_i < 1$ for intermediate particles. We interpolate the material properties, as follows:

$$\rho_i = \bar{\rho}((1 - \varepsilon)\gamma_i + \varepsilon), \quad (9)$$

$$\lambda_i = \bar{\lambda}((1 - \varepsilon)\gamma_i + \varepsilon), \quad (10)$$

$$\mu_i = \bar{\mu}((1 - \varepsilon)\gamma_i + \varepsilon), \quad (11)$$

where $\bar{\rho}$, $\bar{\lambda}$ and $\bar{\mu}$ are the density and Lamé constants of the solid material, respectively, and ε is the small constant to avoid numerical instabilities. We set $\varepsilon = 10^{-5}$ in this study.

To ensure the smoothness of the structures, we apply the particle-wise density filtering which average the fictitious material density of each particle using those of neighboring particles [24]. Let $\phi := [\phi_1, \dots, \phi_{N^{\text{par}}}]$ denote the array of design variables whose component ϕ_i determines the fictitious material density γ_i via density filtering and Heaviside projection. First, we calculate filtered variables $\tilde{\phi}_i$ for each particle, as follows:

$$\tilde{\phi}_i = \frac{\sum_{i'=1}^{N^{\text{par}}} w(\|\mathbf{x}_i - \mathbf{x}_{i'}\|) \phi_{i'}}{\sum_{i'=1}^{N^{\text{par}}} w(\|\mathbf{x}_i - \mathbf{x}_{i'}\|)} \quad (12)$$

where \mathbf{x}_i is the position of the i -th particle and w is the weighting function defined as

$$w(r) := \left(1 - \frac{\min(r, R)}{R}\right)^p, \quad (13)$$

where R is a filter radius and p is a parameter for determining the decay of the weight. Applying the Heaviside projection using the sigmoid function, we obtain

$$\gamma_i = \frac{1}{1 + \exp(-\beta^{\text{to}} \tilde{\phi}_i)}, \quad (14)$$

where β^{to} is a parameter that determines the curvature of the sigmoid function.

3.2. Actuator layout optimization

Next, we formulate the layout optimization problem that determines the self-actuating sub-domains in the material domain of a soft body, which is based on our previous work [24]. Let $\xi_{ij} \in [0, 1]$ be the continuously relaxed two-index variable for assigning the j -th actuation signal to the i -th particle. We define the actuation signal, as follows:

$$a_i(t) = ((1 - \varepsilon)\gamma_i^3 + \varepsilon) \sum_{j=1}^{N^{\text{act}}+1} \xi_{ij} \bar{a}_j(t), \quad (15)$$

where $\bar{a}_{j \leq N^{\text{act}}}$ is the j -th actuation signal and $\bar{a}_{N^{\text{act}}+1} = 0$ for representing no actuation. The coefficient $((1 - \varepsilon)\gamma_i^3 + \varepsilon)$ attenuates the actuation strength on void particles, where the power of γ_i , 3, functions as a penalty to intermediate values of γ_i similar to the solid isotropic material with penalization (SIMP) method [35, 37]. To ensure the smoothness of the layout as well as the structure, we use the particle-wise density filtering technique. Let $\psi := [\psi_{11}, \dots, \psi_{N^{\text{par}}N^{\text{act}}+1}]$ denote the array of design variables whose component ψ_{ij} determines the relaxed multi-index variable ξ_{ij} via density filtering and softmax projection. First, we calculate filtered variable $\tilde{\psi}_{ij}$ for each particle, as follows:

$$\tilde{\psi}_{ij} = \frac{\sum_{i'=1}^{N^{\text{par}}} w(\|\mathbf{x}_i - \mathbf{x}_{i'}\|) \psi_{i'j}}{\sum_{i'=1}^{N^{\text{par}}} w(\|\mathbf{x}_i - \mathbf{x}_{i'}\|)}, \quad (16)$$

where w is defined in Eq. (13). Applying the softmax function to the filtered variables, we obtain

$$\xi_{ij} = \frac{\exp(\beta^{\text{lay}} \tilde{\psi}_{ij})}{\sum_{j'=1}^{N^{\text{act}}+1} \exp(\beta^{\text{lay}} \tilde{\psi}_{ij'})}, \quad (17)$$

where β^{lay} is a parameter for controlling the curvature of the softmax function.

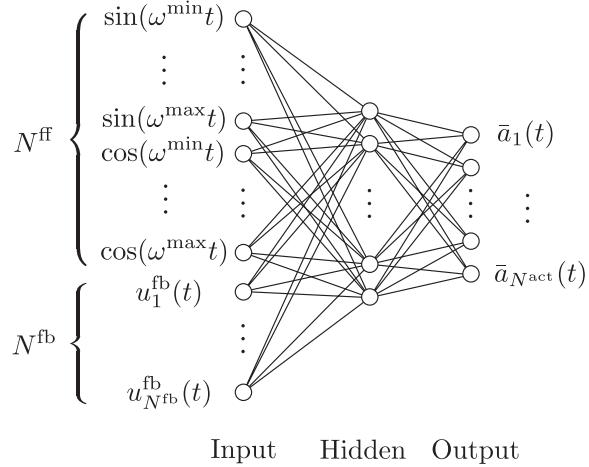


Figure 2: Neural network for the feedback controller. The input signals consist of feedforward and feedback signals where feedforward signals are given as sinusoidal and cosinusoidal functions, and the feedback signals are the terrain height surrounding the soft robots.

3.3. Controller optimization

We now parametrize the actuation signal in the j -th channel, \bar{a}_j . In this study, we use the terrain feature surrounding the soft body to design a feedback controller. Since the terrain feature can be reflected in the boundary condition in Eq. (8), it is reasonable to represent the terrain on the grid node. Let $\mathbf{x}_m^{\text{grid}} = (x_m^{\text{grid}}, y_m^{\text{grid}}, z_m^{\text{grid}})^{\top}$ denote the coordinate of the m -th grid node where the y -axis is taken along with the vertical upward direction. Using the centroid of the material domain, $\mathbf{x}^{\text{cen}}(t)$, defined as

$$\mathbf{x}^{\text{cen}}(t) := \frac{\sum_{i=1}^{N^{\text{par}}} \gamma_i \mathbf{x}_i(t)}{\sum_{i=1}^{N^{\text{par}}} \gamma_i}. \quad (18)$$

We can express the index set corresponding to the terrain surrounding the soft body, as follows:

$$\mathcal{B}^{\text{sur}}(t) := \left\{ m \mid m \in \mathcal{B}, \|\text{proj}(\mathbf{x}_m^{\text{grid}}) - \text{proj}(\mathbf{x}^{\text{cen}}(t))\| \leq \frac{W}{2} \right\}, \quad (19)$$

where W is the diameter to define the surrounding area around the centroid of the soft robot, and proj is the projector of positions onto the plane of $y = 0$ where y -axis be taken along with the vertical upward direction. To define a feedback controller based on this terrain feature, we use the neural network with one hidden layer. Let $\mathbf{w}^{(1)} := [w_{11}^{(1)}, w_{12}^{(1)}, \dots, w_{N^{\text{hidden}} N^{\text{input}}}^{(1)}]$ and $\mathbf{w}^{(2)} := [w_{11}^{(2)}, w_{12}^{(2)}, \dots, w_{N^{\text{act}} N^{\text{hidden}}}^{(2)}]$ be the weights of the first and second layers, respectively. Let $\mathbf{b}^{(1)} := [b_1^{(1)}, \dots, b_{N^{\text{hidden}}}^{(1)}]$ and $\mathbf{b}^{(2)} := [b_1^{(2)}, \dots, b_{N^{\text{act}}}^{(2)}]$ be the bias of the first and second layers, respectively. The neural network that outputs the actuation signal \bar{a}_j from the input u_l through the hidden state \tilde{u}_k is modeled, as follows:

$$\tilde{u}_k(t) = \tanh \left(\sum_{l=1}^{N^{\text{input}}} w_{kl}^{(1)} u_l(t) + b_k^{(1)} \right), \quad (20)$$

$$\bar{a}_j(t) = c^{\text{act}} \tanh \left(\sum_{k=1}^{N^{\text{hidden}}} w_{jk}^{(2)} \tilde{u}_k(t) + b_j^{(2)} \right), \quad (21)$$

for $1 \leq l \leq N^{\text{input}}$, $1 \leq k \leq N^{\text{hidden}}$, $1 \leq j \leq N^{\text{act}}$ where c^{act} is a scaling factor of the actuation signal, N^{input} and N^{hidden} are the numbers of input and hidden nodes, respectively. In addition to the feedback

signals, we also use the feedforward signals as the input of the neural network. Figure 2 illustrates the neural network used for the feedback controller. Let u_l^{ff} and u_l^{fb} respectively denote the feedforward and feedback signals. The input of the neural network is given, as follows:

$$u_l(t) = \begin{cases} u_l^{\text{ff}}(t) & \text{if } 1 \leq l \leq N^{\text{ff}} \\ u_{l-N^{\text{ff}}}^{\text{fb}}(t) & \text{if } N^{\text{ff}} < l \leq N^{\text{ff}} + N^{\text{fb}}, \end{cases} \quad (22)$$

where N^{ff} and N^{fb} are the number of feedforward and feedback signals, respectively. The feedforward signals consist of sinusoidal and cosinusoidal functions as

$$u_l^{\text{ff}}(t) = \begin{cases} \sin(\omega_l t) & \text{if } 1 \leq l \leq \lfloor \frac{N^{\text{ff}}}{2} \rfloor \\ \cos(\omega_{l-\lfloor N^{\text{ff}}/2 \rfloor} t) & \text{if } \lfloor \frac{N^{\text{ff}}}{2} \rfloor < l \leq N^{\text{ff}}, \end{cases} \quad (23)$$

where

$$\omega_l := (\omega^{\max} - \omega^{\min}) \frac{l-1}{\lceil N^{\text{ff}}/2 \rceil - 1} + \omega^{\min} \quad (24)$$

with the maximum and minimum angular frequencies, denoted by ω^{\max} and ω^{\min} , respectively. The feedback signal is defined using the terrain feature surrounding the soft body, $\mathcal{B}^{\text{sur}}(t)$, as follows:

$$u_l^{\text{fb}}(t) = \tanh \left(c^{\text{fb}} \left(\mathbf{x}^{\text{cen}}(t) \cdot \mathbf{e}_y - y_{(\mathcal{B}^{\text{sur}}(t))_l}^{\text{grid}} - y^{\text{offset}} \right) \right), \quad (25)$$

where \mathbf{e}_y is the basis vector along with the y -axis, c^{fb} is a scaling factor, y^{offset} is the offset and $(\mathcal{B}^{\text{sur}}(t))_l$ is the l -th element of $\mathcal{B}^{\text{sur}}(t)$. By this definition, the number of feedback signals coincides the number of elements of $\mathcal{B}^{\text{sur}}(t)$, that is, $N^{\text{fb}} = |\mathcal{B}^{\text{sur}}(t)|$. However, especially for three-dimensional problems, the number of feedback signals tends to be so large depending on the number of nodes in the grid. Hence, for three-dimensional problems, we select representative nodes at every three nodes along the x and z -axes from the index set \mathcal{B}^{sur} , forming a new index set $\tilde{\mathcal{B}}^{\text{sur}}$. Then, we take the convolution of 3×3 nodes to reduce the number of signals, as follows:

$$\text{For } m \in \tilde{\mathcal{B}}^{\text{sur}}(t), \quad \tilde{y}_m^{\text{grid}} = \frac{1}{|\mathcal{I}_m^{\text{neigh}} \cap \mathcal{B}^{\text{sur}}(t)|} \sum_{m' \in \mathcal{I}_m^{\text{neigh}} \cap \mathcal{B}^{\text{sur}}(t)} y_{m'}^{\text{grid}}, \quad (26)$$

where $\mathcal{I}_m^{\text{neigh}}$ represents the index set of 3×3 nodes neighboring the m -th node. The feedback signal is given by replacing $y_{(\mathcal{B}^{\text{sur}}(t))_l}^{\text{grid}}$ in Eq. (25) with $\tilde{y}_{(\tilde{\mathcal{B}}^{\text{sur}}(t))_l}^{\text{grid}}$, as follows:

$$u_l^{\text{fb}}(t) = \tanh \left(c^{\text{fb}} \left(\mathbf{x}^{\text{cen}}(t) \cdot \mathbf{e}_y - \tilde{y}_{(\tilde{\mathcal{B}}^{\text{sur}}(t))_l}^{\text{grid}} - y^{\text{offset}} \right) \right). \quad (27)$$

Figure 3 illustrates the concept of the convolution of terrain features. This is a kind of the convolutional neural network (CNN) and we can also use more sophisticated CNNs for this purpose. Design variables for the neural network is summarized as $\mathbf{w} := [\mathbf{w}^{(1)}, \mathbf{b}^{(1)}, \mathbf{w}^{(2)}, \mathbf{b}^{(2)}]$.

3.4. Objective function

Next, we formulate the objective function. Let L denote the target travel distance during a time duration T . The objective function \mathcal{F} for a soft body to achieve the target travel distance keeping the posture is formulated, as follows:

$$\mathcal{F}(\phi, \psi, \mathbf{w}) := \min \left(1 - \frac{\int_0^T \mathbf{v}^{\text{cen}}(t) \cdot \mathbf{e}_x dt}{TL}, 0 \right)^2 + \frac{w_\theta}{T} \int_0^T \left\| \frac{\boldsymbol{\theta}(t)}{\dot{\theta}} \right\|^2 dt, \quad (28)$$

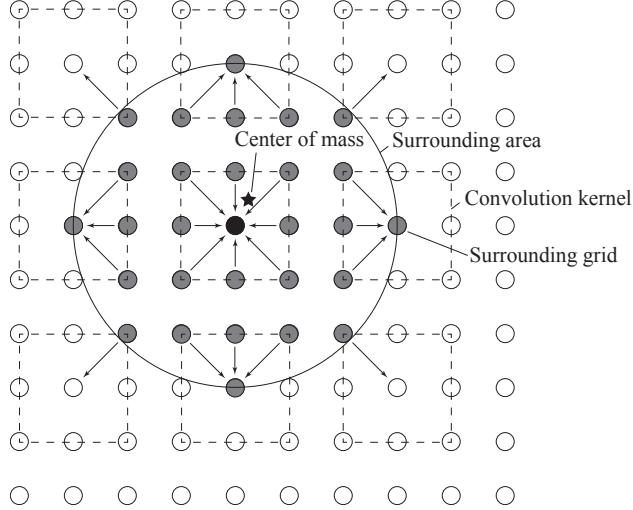


Figure 3: Conceptual diagram of convolution of terrain features. The star sign represents the center of mass of the soft robot and the black circle represents the grid node corresponding to the center of mass. A large circle defines the surrounding area and the terrain heights of the nodes in the circle (i.e., gray nodes whose indices form $\mathcal{B}^{\text{sur}}(t)$) are used as feedback signals. Dashed boxes represent 3×3 neighboring nodes whose terrain heights are convolved into the center node of each box, whose index forms $\tilde{\mathcal{B}}^{\text{sur}}(t)$.

where w_θ is a weighting coefficient and \mathbf{v}^{cen} is the velocity of the centroid defined as

$$\mathbf{v}^{\text{cen}}(t) := \frac{\sum_{i=1}^{N^{\text{par}}} \gamma_i \mathbf{v}_i(t)}{\sum_{i=1}^{N^{\text{par}}} \gamma_i}, \quad (29)$$

where $\boldsymbol{\theta}$ is the rotation vector and $\bar{\theta}$ is a normalization parameter. Note that the velocity \mathbf{v}_i and the rotation vector $\boldsymbol{\theta}$ implicitly depend on the design variables ϕ , ψ and \mathbf{w} through the forward problem. In the following, we explain how to calculate the rotation vector $\boldsymbol{\theta}$. Let \mathbf{J} denote the inertia matrix around the centroid. The angular velocity vector $\boldsymbol{\omega}$ is then given as

$$\boldsymbol{\omega}(t) = \mathbf{J}^{-1}(t) \sum_{i=1}^{N^{\text{par}}} \rho_i (\mathbf{x}_i(t) - \mathbf{x}^{\text{cen}}(t)) \times (\mathbf{v}_i(t) - \mathbf{v}^{\text{cen}}(t)). \quad (30)$$

The unit quaternion $\mathbf{q}(t)$ describing the posture of the soft body is given as

$$\mathbf{q}(t) = \mathbf{q}(0) + \frac{1}{2} \int_0^t \boldsymbol{\omega}(t') \mathbf{q}(t') dt', \quad (31)$$

where $\boldsymbol{\omega}(t')\mathbf{q}(t')$ is the product of quaternions. Then, the rotation vector is calculated by the quaternion, as follows:

$$\boldsymbol{\theta}(t) = \frac{2 \tan^{-1} \left(\frac{\sqrt{q_1(t)^2 + q_2(t)^2 + q_3(t)^2}}{q_0(t)} \right)}{\sqrt{q_1(t)^2 + q_2(t)^2 + q_3(t)^2}} [q_1(t), q_2(t), q_3(t)]^\top, \quad (32)$$

where q_0, q_1, q_2, q_3 are components of the quaternion \mathbf{q} . By minimizing the objective function \mathcal{F} , we obtain the optimized structure, actuator layout, and neural network-based controller of a soft body for a prescribed terrain given by y_m^{grid} .

Since we would like to design a soft body which can travel under terrain uncertainty, we prepare dataset of terrains and *train* the soft body using the dataset based on the machine learning approach. Let $\mathcal{F}^{(n)}$ be

the objective function for the n -th terrain data. Then, the optimization problem to train the soft body by N^{data} training data is formulated, as follows:

$$\underset{\phi, \psi, w}{\text{minimize}} \quad \frac{1}{N^{\text{data}}} \sum_{n=1}^{N^{\text{data}}} \mathcal{F}^{(n)}(\phi, \psi, w) + \frac{1}{2} \alpha \|w\|^2 \quad (33)$$

$$\text{subject to: } \mathcal{C}^{\text{to}}(\phi) \leq \bar{C}^{\text{to}} \quad (34)$$

$$\mathcal{C}^{\text{lay}}(\psi) \leq \bar{C}^{\text{lay}}, \quad (35)$$

where α is a coefficient for L2 regularization of the neural network and \mathcal{C}^{to} and \mathcal{C}^{lay} are the constraint functions defined as

$$\mathcal{C}^{\text{to}}(\phi) := \frac{4}{N^{\text{par}}} \sum_{i=1}^{N^{\text{par}}} \gamma_i(\phi_i)(1 - \gamma_i(\phi_i)) \quad (36)$$

$$\mathcal{C}^{\text{lay}}(\psi) := \frac{N^{\text{act}} + 1}{N^{\text{par}} N^{\text{act}}} \sum_{i=1}^{N^{\text{par}}} \left(1 - \sum_{j=1}^{N^{\text{act}} + 1} \xi_{ij}(\psi_{ij})^2 \right), \quad (37)$$

for ensuring the binarization of the material density γ_i and relaxed two-index variable ξ_{ij} by setting the upper bounds \bar{C}^{to} and \bar{C}^{lay} to a small value. Here, we used the notation $\gamma_i(\phi_i)$ and $\xi_{ij}(\psi_{ij})$ to explicitly show that they depend on design variables ϕ_i and ψ_{ij} through the particle-wise filtering and Heaviside projection/softmax projection, respectively. The function $\mathcal{C}^{\text{to}}(\phi) \in [0, 1]$ has the minimum value of zero when γ_i takes either zero or one for all particles while it has the maximum value of one when $\gamma_i = 0.5$ for all particles. The function $\mathcal{C}^{\text{lay}}(\psi) \in [0, 1]$ has the minimum value of zero when $\xi_{ij} = 1$ for a value of j and $\xi_{ij'} = 0$ for all $j' \neq j$ for all particles while it has the maximum value of one when $\xi_{ij} = 1/(N^{\text{act}} + 1)$ for all j and all particles. That is, \mathcal{C}^{to} and \mathcal{C}^{lay} represent the ratio of intermediate values of γ_i and ξ_{ij} , respectively.

3.5. Dataset

For training a soft body, we prepare terrain dataset by randomly sampling terrains from Gaussian process. Let a box $[0, L_x] \times [0, L_y] \times [0, L_z]$ denote the simulation environment with the coordinate $\mathbf{x} = [x, y, z]$. We take y -axis along with the vertical upward direction as in Sec. 3.3. Let \mathcal{K} be the Gaussian kernel defined as

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') := \exp \left(-\frac{(x - x')^2 + (z - z')^2}{2\varrho^2} \right), \quad (38)$$

where ϱ is a hyperparameter determining the radius of the kernel. We then sample a function describing the height of terrains from the Gaussian process $\mathcal{GP}(\bar{h}, \mathcal{K})$ with the constant mean function \bar{h} and the kernel function \mathcal{K} , as follows:

$$h(x, z) \sim \mathcal{GP}(\bar{h}, \varsigma \mathcal{K}), \quad (39)$$

where ς is a hyperparameter for scaling the height difference in a terrain. The function $h(x, z)$ gives the height of a terrain at a horizontal coordinate (x, z) . For performing the forward simulation using each terrain data, we first have to put a soft body on an initial position, which should be fixed for all terrain data to eliminate the influence of initial positions. However, since the function $h(x, z)$ gives the random height all over the environment, it is difficult to determine an initial position available for all data. Thus, we fix the height of terrains on the area satisfying $x \leq \bar{x}$ to \bar{h} and set the initial position on the area. To make the terrain continuously connect to this fixed area, we modify the function $h(x, z)$, as follows:

$$\tilde{h}(x, z) = h(x, z) - \frac{h(L_x, z) - h(\bar{x}, z)}{L_x - \bar{x}} (x - \bar{x}) - h(\bar{x}, z) + \bar{h}. \quad (40)$$

Algorithm 1: Optimization algorithm

Input : Initial design variables ϕ^0, ψ^0 and \mathbf{w}^0 , penalty coefficient τ^0 .
Output: Optimized structure $[\gamma_1, \dots, \gamma_{N\text{par}}]$, actuator layout $[\xi_{11}, \dots, \xi_{N\text{par}N\text{act}+1}]$, neural network parameters \mathbf{w}

```

1 Divide  $N^{\text{data}}$  into  $N^{\text{batch}}$  batches  $\{\mathcal{I}_n^{\text{batch}}\}_{n=1}^{N^{\text{batch}}}$  randomly.
2 Set  $s \leftarrow 0, n \leftarrow 1, \kappa \leftarrow \mathbf{0}, \tau \leftarrow \tau^0$ .
3 Set  $\text{flag} \leftarrow \text{True}$ .
4 while  $\text{flag}$  do
5   Compute  $\mathcal{L}_n(\phi^s, \psi^s, \mathbf{w}^s, \kappa, \tau)$  and its gradients.
6   Set  $\mathcal{L}^s \leftarrow \mathcal{L}_n$ .
7   if  $\left| \sum_{s'=s-N^{\text{batch}}+1}^s \mathcal{L}^{s'} - \sum_{s'=s-2N^{\text{batch}}+1}^{s-N^{\text{batch}}} \mathcal{L}^{s'} \right| < \epsilon$  then
8     if  $\mathcal{C}^{\text{to}} \leq \bar{\mathcal{C}}^{\text{to}}$  and  $\mathcal{C}^{\text{lay}} \leq \bar{\mathcal{C}}^{\text{lay}}$  then
9       Set  $\text{flag} \leftarrow \text{False}$ .
10      Return  $\{\gamma_i(\phi_i^s)\}_i, \{\xi_{ij}(\psi_i^s)\}_{ij}, \mathbf{w}^s$ .
11    else
12      Update Lagrange multiplier  $\kappa$  and penalty coefficient  $\tau$ .
13    Update design variables to  $\phi^{s+1}, \psi^{s+1}$  and  $\mathbf{w}^{s+1}$  by Adam.
14    if  $n = N^{\text{batch}}$  then
15      Divide  $N^{\text{data}}$  into  $N^{\text{batch}}$  batches  $\{\mathcal{I}_n^{\text{batch}}\}_{n=1}^{N^{\text{batch}}}$  randomly.
16      Set  $n \leftarrow 0$ .
17    Set  $s \leftarrow s + 1, n \leftarrow n + 1$ .
18 end

```

The modified function $\tilde{h}(x, z)$ satisfies $\tilde{h}(\bar{x}, z) = \tilde{h}(L_x, z) = \bar{h}$. We eventually obtain an index set describing grid nodes on the ground for the boundary condition in Eq. (8), as follows:

$$\mathcal{B} = \left\{ m \mid y_m^{\text{grid}} = \left\lfloor \frac{\tilde{h}(x_m^{\text{grid}}, z_m^{\text{grid}})}{\Delta_y} \right\rfloor \Delta_y \right\}, \quad (41)$$

where Δ_y is the interval between grid nodes along with the y -axis. The floor function maps the continuous height onto the discrete grid. We also prepare test dataset as well as training dataset.

3.6. Optimization algorithm

To solve the optimization problem formulated in Sec. 3.4, we use the augmented Lagrangian method [38], which converts a constrained problem to an unconstrained problem. As an optimizer, we use the Adam [39], a kind of stochastic gradient methods, with mini-batch learning. That is, we pick up partial N^{batch} data at an iteration to evaluate the objective function and its gradients and to update the design variables.

Let $\mathcal{I}_n^{\text{batch}}$ denote the index set of training data in the n -th batch with the size N^{batch} . We assume that all data N^{data} can be divided into N^{batch} batches without residue satisfying $\bigcup_{n=1}^{N^{\text{batch}}} \mathcal{I}_n^{\text{batch}} = \{1, 2, \dots, N^{\text{data}}\}$. The augmented Lagrangian \mathcal{L} for the n -th batch is formulated, as follows:

$$\begin{aligned}
& \mathcal{L}_n(\phi, \psi, \mathbf{w}, \kappa, \tau) \\
&:= \frac{1}{|\mathcal{I}_n^{\text{batch}}|} \sum_{n' \in \mathcal{I}_n^{\text{batch}}} \mathcal{F}^{(n')} + \frac{1}{2} \alpha \|\mathbf{w}\|^2 \\
&\quad - \kappa^{\text{to}} \max(\mathcal{C}^{\text{to}}(\phi) - \bar{\mathcal{C}}^{\text{to}}, 0) + \frac{1}{2} \tau^{\text{to}} \max(\mathcal{C}^{\text{to}}(\phi) - \bar{\mathcal{C}}^{\text{to}}, 0)^2 \\
&\quad - \kappa^{\text{lay}} \max(\mathcal{C}^{\text{lay}}(\psi) - \bar{\mathcal{C}}^{\text{lay}}, 0) + \frac{1}{2} \tau^{\text{lay}} \max(\mathcal{C}^{\text{lay}}(\psi) - \bar{\mathcal{C}}^{\text{lay}}, 0)^2,
\end{aligned} \quad (42)$$

where $\boldsymbol{\kappa} := [\kappa^{\text{to}}, \kappa^{\text{lay}}]$ is the Lagrange multiplier and $\boldsymbol{\tau} := [\tau^{\text{to}}, \tau^{\text{lay}}]$ is the penalty coefficient. Since the average of the augmented Lagrangians for all batches corresponds to the augmented Lagrangian for all data, that is,

$$\begin{aligned} & \frac{1}{N^{\text{batch}}} \sum_{n=1}^{N^{\text{batch}}} \mathcal{L}_n \\ &= \frac{1}{N^{\text{data}}} \sum_{n'=1}^{N^{\text{data}}} \mathcal{F}^{(n')} + \frac{1}{2} \alpha \|\boldsymbol{w}\|^2 \\ &\quad - \kappa^{\text{to}} \max(\mathcal{C}^{\text{to}}(\boldsymbol{\phi}) - \bar{C}^{\text{to}}, 0) + \frac{1}{2} \tau^{\text{to}} \max(\mathcal{C}^{\text{to}}(\boldsymbol{\phi}) - \bar{C}^{\text{to}}, 0)^2 \\ &\quad - \kappa^{\text{lay}} \max(\mathcal{C}^{\text{lay}}(\boldsymbol{\psi}) - \bar{C}^{\text{lay}}, 0) + \frac{1}{2} \tau^{\text{lay}} \max(\mathcal{C}^{\text{lay}}(\boldsymbol{\psi}) - \bar{C}^{\text{lay}}, 0)^2, \end{aligned} \quad (43)$$

the stochastic gradient approach works to solve the problems in Eqs. (33)–(35) by the augmented Lagrangian method and mini-batch learning. We describe the optimization algorithm in Algorithm 1. We implemented this algorithm using Python and Taichi [40, 41], an open-source programming language which supports parallel computing on GPU and provides an automatic differentiation (AD) scheme. We used AD to compute the gradient in line 5 in the algorithm.

4. Numerical experiments

We now present two numerical experiments, where we design a soft body that moves toward x -direction, which we named *Walker*, in two- and three-dimensions. We executed our algorithms on the GPU (NVIDIA A100 40GB) for all examples.

4.1. Hyperparameter settings

We first describe hyperparameters using the same values across all numerical experiments. Hyperparameters set on specific problems are provided in the respective examples.

4.1.1. Material property

The soft body consisted of a soft silicone rubber with the density $\bar{\rho} = 1000 \text{ kg/m}^3$, the Young's modulus $E = 0.1 \text{ MPa}$ and the Poisson ratio $\nu = 0.4$, which have the relationship with the Lamé constants that

$$\bar{\lambda} = \frac{E\nu}{(1+\nu)(1-2\nu)} \quad (44)$$

$$\bar{\mu} = \frac{E}{2(1+\nu)}. \quad (45)$$

We set the small constant $\varepsilon = 10^{-5}$ for numerical stability.

4.1.2. Forward problem settings

The computational domain representing simulation environment consisted of a square and a cube in 2D and 3D problems, respectively, with a length of $L_x = L_y = L_z = 1 \text{ m}$. We discretized the domain by a grid with equal spacing $1.25 \times 10^{-2} \text{ m}$. We placed particles for the MPM at $6.25 \times 10^{-3} \text{ m}$ intervals, which is half the length of the grid spacing. We used $\eta = 1 \times 10^2 \text{ Pa} \cdot \text{s}$ for the damping parameter to mitigate high-frequency oscillations. The simulation time was set to $T = 1 \text{ s}$ which we divided with a time step of $\Delta_t = 1 \times 10^{-4} \text{ s}$. The gravity acceleration was set to $\boldsymbol{g} = [0, -9.8 \text{ m/s}^2, 0]^\top$.

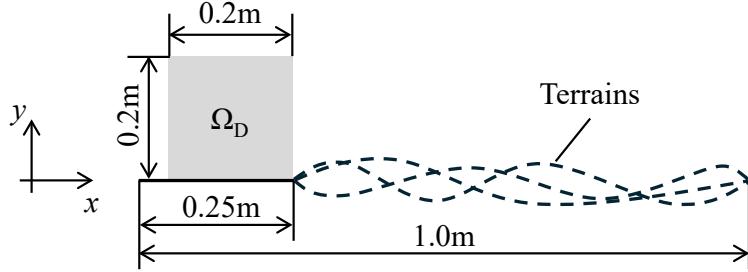


Figure 4: Problem setting for the 2D walker. The gray square represents the design domain of a soft robot and dashed line illustrates the schematic diagram of randomly generated terrains. The soft robot is optimized so that the gray square can travel toward the right direction.

4.1.3. Hyperparameters for the neural network

We initialized neural network parameters for the controller by sampling from the Gaussian distribution \mathcal{N} , based on the Xavier initialization scheme [42], as follows:

$$w_{kl}^{(1)} \sim \mathcal{N} \left(0, \sqrt{\frac{2}{N^{\text{hidden}} + N^{\text{input}}}} \right) \quad (46)$$

$$b_k^{(1)} = 0 \quad (47)$$

$$w_{jk}^{(2)} \sim \mathcal{N} \left(0, \sqrt{\frac{2}{N^{\text{act}} + 1 + N^{\text{hidden}}}} \right) \quad (48)$$

$$b_j^{(2)} = 0, \quad (49)$$

which serves for the outputs and gradients of neural networks to flow effectively in both forward and back propagation. We set $\alpha = 0.01$, the regularization coefficient for the neural network. We prepared $N^{\text{ff}} = 56$ feedforward signals with the maximum and minimum angular frequencies configured to $\omega^{\max} = 8 \times 2\pi$ and $\omega^{\min} = 2 \times 2\pi$, respectively. We configured the offset of feedback signals as $y^{\text{offset}} = \mathbf{x}^{\text{cen}}(0) \cdot \mathbf{e}_y - \bar{h}$, which calibrated the feedback signals at an initial time to zero. The number of hidden nodes was determined by $N^{\text{hidden}} = \lfloor (N^{\text{input}} + N^{\text{act}} + 1)/2 \rfloor$. The actuation signal is scaled by $c^{\text{act}} = 2 \times 10^4$ Pa.

4.1.4. Hyperparameters for optimization

We set initial design variables for the structure and actuator layout as $\phi = \mathbf{0}$ and $\psi = \mathbf{0}$. We also set $R = 9.375 \times 10^{-3}$ m, which is one and a half times than particle intervals, $p = 2$, $\beta^{\text{to}} = 4$ and $\beta^{\text{lay}} = 4$ for the filtering and projection of design variables to fictitious density $\{\gamma_i\}$ and relaxed two-index variables $\{\xi_{ij}\}$. For the objective function, we set $\bar{\theta} = \pi/2$, which is the normalization parameter of the angle describing posture of the soft body. As for the augmented Lagrangian method, we assigned $\tau^0 = [\tau^{\text{to}}, \tau^{\text{lay}}] = [0.001, 0.001]$ and these values were updated in line 12 of Algorithm 1 based on the same strategy with our previous work [24]. We set the learning rate of Adam to 0.02.

4.2. 2D walker

We first designed a soft robot that moved toward x -direction in two-dimensions. We configured the fixed design domain Ω_D as a 0.2×0.2 m 2 square on a flat ground as shown in Fig. 4. We set the number of actuators as $N^{\text{act}} = 4$. For terrain dataset, we assigned $\varsigma = 0.02$ m and $\varrho = 0.2$ m, respectively. We empirically set $w_\theta = 1$ for the weighting coefficient in the objective function (28). We set $c^{\text{fb}} = 1/\varsigma = 50$ m $^{-1}$, the parameter for scaling the feedback signals in Eq. (25), which normalized the feedback signals independent of the terrain heights.

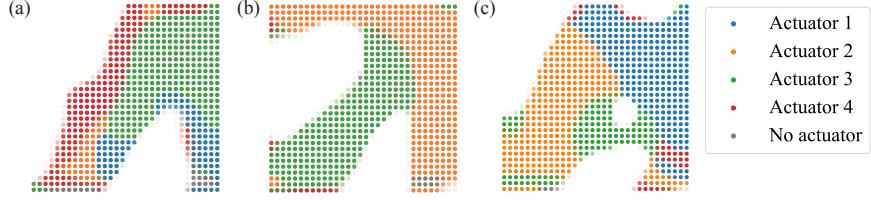


Figure 5: Optimized structures and actuator layouts for various settings of W . (a) $W = 0$, (b) $W = 0.3$ and (c) $W = 0.4$.

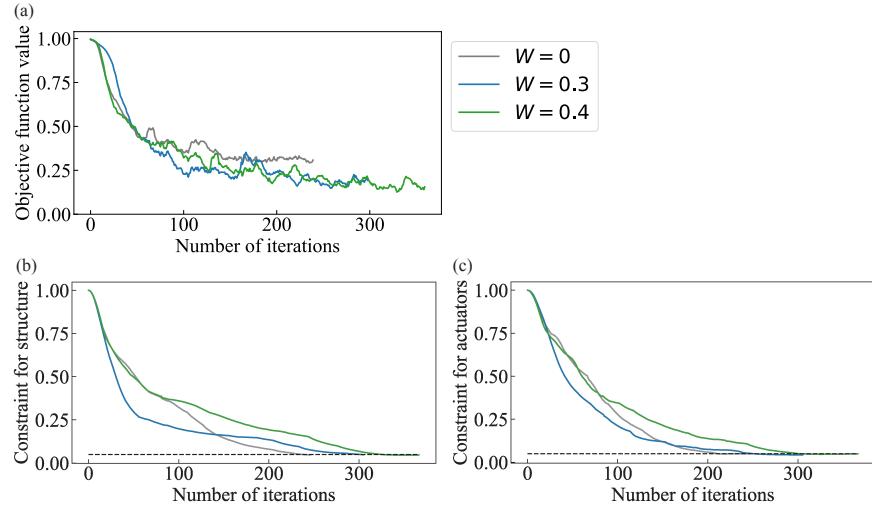


Figure 6: Trajectories of (a) objective function, (b) constraint function for binarization of structure and (c) constraint function for binarization of actuator layout during optimization with respect to the surrounding area width. The objective function values are plotted with a moving average taken every $N^{\text{data}}/N^{\text{batch}} (= 8)$ iterations. Dashed lines in (b) and (c) represents the prescribed upper bounds of the constraint functions.

4.2.1. Effect of size of surrounding area

First, we examined the effect of the size of the surrounding area, W , to be fed into the feedback controller. For the target travel distance, we set $L = 0.7$ m. We prepared 32 ($= N^{\text{data}}$) training data and also prepared the same number of test data.

Figure 5 illustrates the optimized structures and actuation layouts obtained for various settings of the surrounding area W . The case with $W = 0$ means the soft robot employs no feedback controller and just uses feedforward signals u_l^{ff} in Eq. (22). All optimized structures in Fig. 5 have a common feature of two leg-like substructures while detailed parts differ. Actuator layouts on the structures also share similarities; the left and right *legs* are equipped with different actuators. These features also appeared in the previous study [24] and seem to be essential to walk.

Figure 6 shows the trajectories of the objective function \mathcal{F} and constraint functions for binarization of material density and relaxed two-index variables, \mathcal{C}^{to} and \mathcal{C}^{lay} , over the optimization iterations where the objective function was plotted with a moving average taken every $N^{\text{data}}/N^{\text{batch}} (= 8)$ iteration. For all settings of W values, the objective function decreased from 1.0 and stayed around 0.25 with oscillation while constraint functions smoothly reached the upper bounds represented by the dashed lines to satisfy the constraints. Oscillations in the objective function values stem from the stochastic gradient method where the design variables were updated by using different training data at every iteration. No such oscillation was observed for the constraint functions due to the independence of the constraint function from the training dataset. These figures show that our algorithm could decrease the objective function in a stochastic manner while dealing with the constraint functions for structural and actuator layout optimization. The objective

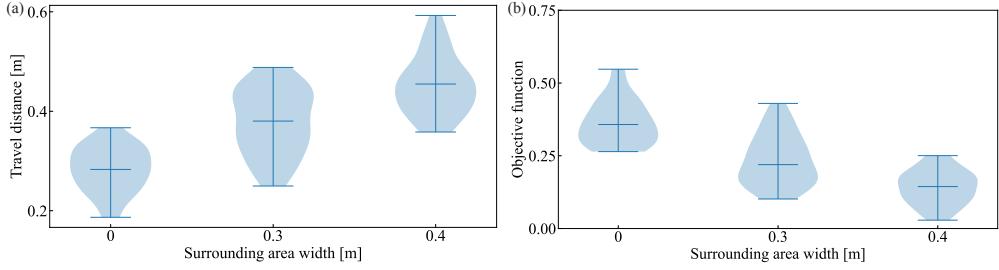


Figure 7: Violin plots illustrating (a) travel distances and (b) objective function values of the optimized soft robots for the test dataset with respect to the surrounding area width W .

function values in the cases of $W = 0.3$ and 0.4 were lower than that of $W = 0$ on average, which implies that the feedback controller contributes to soft robots walking on varying terrains more stably.

Next, we discussed the result using the test dataset which was not used in the training (optimization) procedure. Figure 7 shows violin plots illustrating the travel distances and objective function values of the optimized soft robots in Fig. 5, evaluated for 32 test data. Figure 7(a) indicates that the travel distances for $W = 0.3$ and 0.4 were larger on average than that for $W = 0$, which suggests that the feedback controller contributes to improving the travel distance. In terms of the objective function, which includes the term for keeping the posture in addition to increasing the travel distance, the cases of $W = 0.3$ and 0.4 resulted in better performance on average than the case of $W = 0$ as well. The objective function values evaluated using the test dataset roughly range around 0.25, closely aligning with those obtained from the training dataset. This confirmation assures us that the training process did not result in overfitting.

To analyze the differences in results by various values of W in more detail, we provide the behaviors of optimized soft robots on two particular test terrains. Figure 8 shows the behaviors of optimized soft robots on the test terrain where the largest objective function was observed for the soft robot equipped with a feedforward controller (i.e., the soft robot optimized by setting $W = 0$). The red and blue regions respectively represent areas undergoing expansion and contraction through actuation. The black line represents the terrain on which the orange segment is the range used as the input of the controller. In Fig. 8, the soft robot optimized by setting $W = 0$ required a time to go over steps during $t = 0.35\text{--}1.0$ s while the soft robots obtained by setting $W = 0.3$ and 0.4 advanced over the steps smoothly. This is because the controller could determine the actuation using the terrain feature in the case of $W = 0.3$ and 0.4 . Figure 9 illustrates actuation signals applied to the soft robot optimized by setting $W = 0.4$, that is, actuation signals applied to exhibit the dynamics of the rightmost column of Fig. 8. Each line represents the deviation of each actuation signal from its mean value of all test data while each dashed line represents the actuation signal. We observe that the large negative deviation for the actuator 2 and the large positive deviation for the actuator 3 during $t = 0.35\text{--}0.65$ s, which enables the widening of the gap between the front and rear legs, contributing to advancing over the steps. Figure 10 illustrates the behaviors of optimized soft robots on the test terrain where the smallest objective function was observed for the soft robot equipped with a feedforward controller. The soft robot optimized by setting $W = 0$ lost the balance at a step after $t = 0.65$ s, which improved the objective function by increasing the travel distance. On the other hand, the soft robots obtained by setting $W = 0.3$ and 0.4 advanced on the step keeping the balance to time $t = 1.0$ s. Thus, we observed the soft robots equipped with the feedback controller could travel on varying terrains more smoothly. Figure 11 shows actuation signals applied to exhibit the dynamics of the rightmost column of Fig. 10. We observe the large positive deviation for the actuator 2 and the large negative deviation for the actuator 3, which implies that the soft robots are not taking larger steps compared to them on other terrains because of the downhill.

4.2.2. Effect of dataset size

Next, we examined the effect of training dataset size. We set the target travel distance as $L = 0.7$ m. For the size of the surrounding area, we set $W = 0.4$ m.

Figure 12 illustrates the optimized structures and actuation layouts obtained using training data with

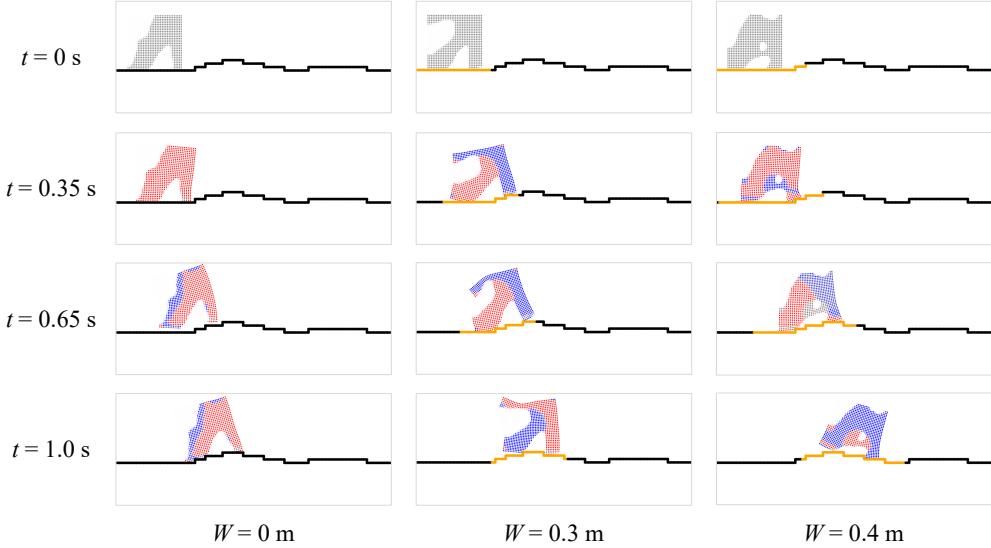


Figure 8: Dynamic behaviors of the three optimized soft robots for the test terrain on which the largest objective function was observed among the test dataset for the soft robot equipped with a feedforward controller. The red and blue represent the expansion and contraction by actuation, respectively. The black line represents the terrain where the orange segment is the range used as the input of the controller.

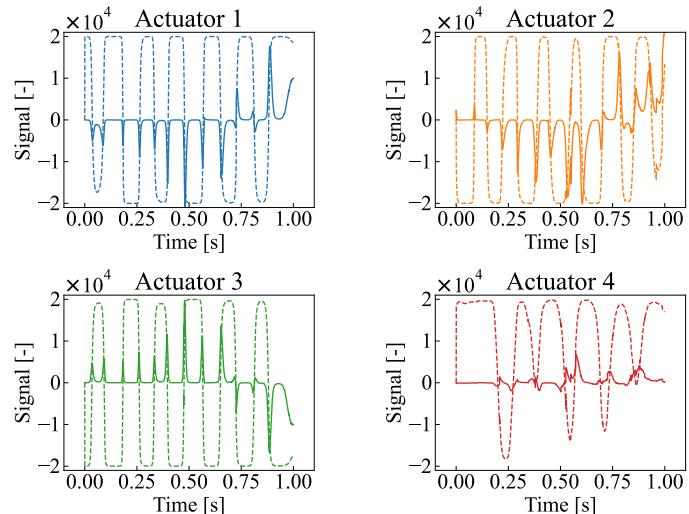


Figure 9: Signals of the respective actuators over the simulation duration. Each line represents the deviation of each actuation signal from its mean value of all test data. Each dashed line represents the actuation signal. Positive and negative actuation signals represent expansion and contraction, respectively.

the size of $N^{\text{data}} = 16$ and 64 . These structures have a common feature of two leg-like substructures as in Fig. 5. Actuator layouts on the structures also share similarities to the results in Fig. 5; the left and right legs are equipped with different actuators. Especially, the optimized structures in Fig. 5(c) for $N^{\text{data}} = 32$ and Fig. 12(b) for $N^{\text{data}} = 64$ agreed well, which implied that 32 training data was enough to train the walker on one-dimensional terrains.

Figure 13 illustrates the history of the objective function \mathcal{F} and constraint functions for binarization of material density and relaxed two-index variables, \mathcal{C}^{to} and \mathcal{C}^{lay} , during the optimization iterations where the

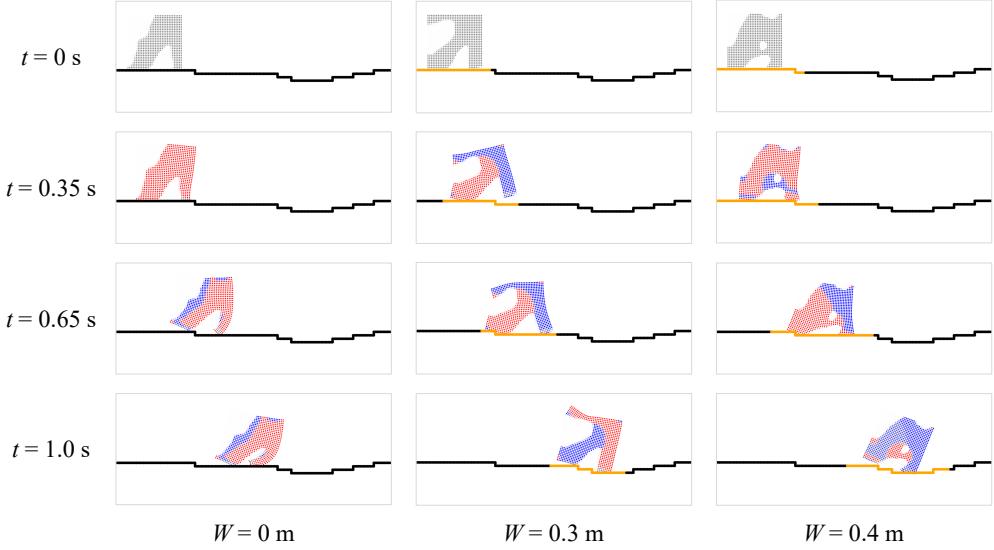


Figure 10: Dynamic behaviors of the three optimized soft robots for the test terrain on which the smallest objective function was observed among the test dataset for the soft robot equipped with a feedforward controller. The red and blue represent the expansion and contraction by actuation, respectively. The black line represents the terrain where the orange segment is the range used as the input of the controller.

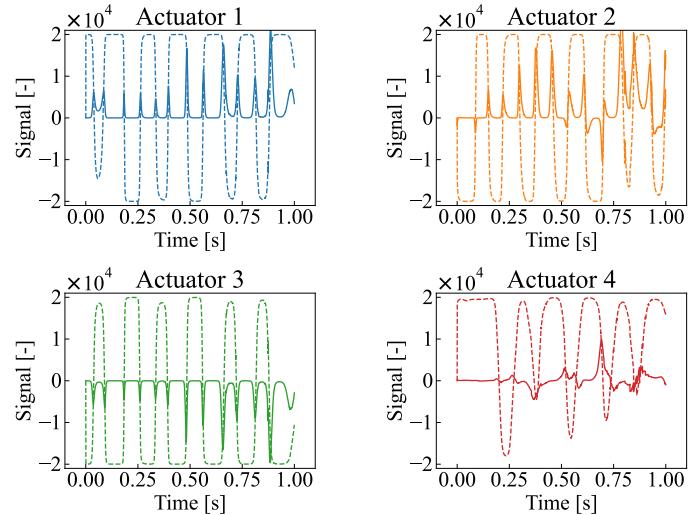


Figure 11: Signals of the respective actuators over the simulation duration. Each line represents the deviation of each actuation signal from its mean value of all test data. Each dashed line represents the actuation signal. Positive and negative actuation signals represent expansion and contraction, respectively.

objective function was plotted with a moving average taken every $N^{\text{data}}/N^{\text{batch}}$ iteration. We also included the results for $N^{\text{data}} = 32$ in the previous subsection for comparison. For all cases, the objective function successfully decreased from 1.0 and stayed below 0.25 with oscillation while constraint functions smoothly decreased to satisfy the constraints. We observed no significant differences among three cases of $N^{\text{data}} = 16$, 32, and 64.

Figure 14 shows a violin plot illustrating the travel distances and objective function values evaluated using 32 test data for the soft robots optimized by 16, 32, and 64 training data. There are no significant

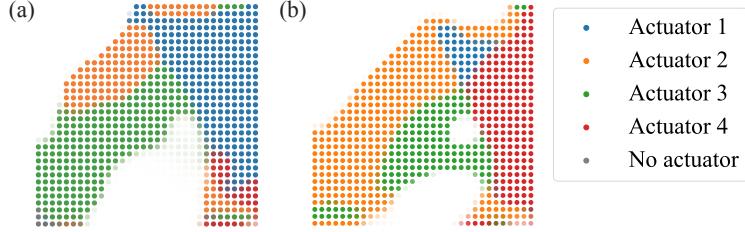


Figure 12: Optimized structures and actuator layouts using training data with the size of (a) $N^{\text{data}} = 16$ and (b) $N^{\text{data}} = 64$.

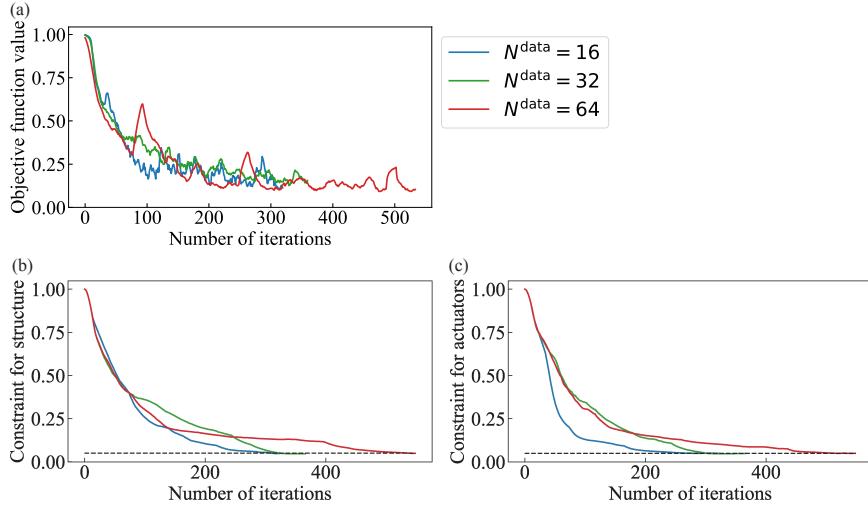


Figure 13: Trajectories of (a) objective function, (b) constraint function for binarization of structure, and (c) constraint function for binarization of actuator layout during optimization with respect to the training dataset size. The objective function values are plotted with a moving average taken every $N^{\text{data}}/N^{\text{batch}}$ iterations. Dashed lines in (b) and (c) represent the prescribed upper bounds of the constraint functions.

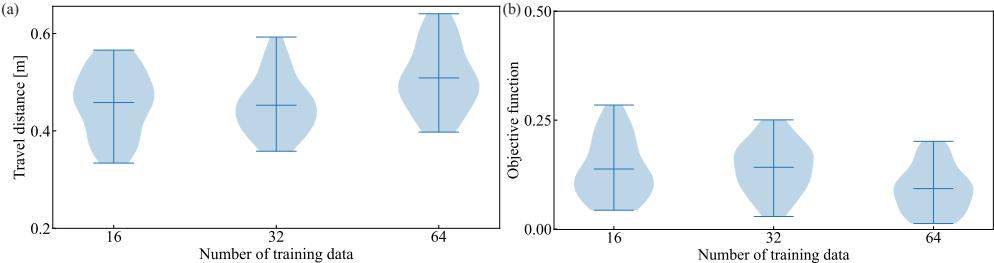


Figure 14: Violin plots illustrating (a) travel distances and (b) objective function values of the optimized soft robots for the test dataset with respect to the training dataset size.

differences in travel distances and objective function values in these three cases while large data slightly improved these metrics. As a result, we confirmed that a few dozen pieces of data suffice to train the walker on one-dimensional terrains. The objective function values evaluated using the test dataset are almost below 0.25, which we again confirmed that the training process did not result in overfitting.

4.3. 3D walker

Finally, we designed a soft body that moved toward x -direction in three dimensions. We configured the fixed design domain Ω_D as a $0.15 \times 0.15 \times 0.15 \text{ m}^3$ cube on a flat ground as shown in Fig. 15. We

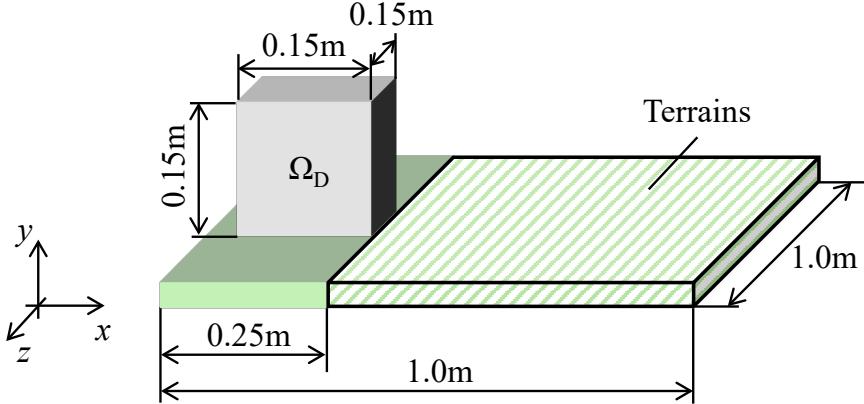


Figure 15: Problem setting for the 3D walker. The gray square represents the design domain of a soft robot and hatched box illustrates an area where terrains are randomly generated. The soft robot is optimized so that the gray square can travel toward the right direction.

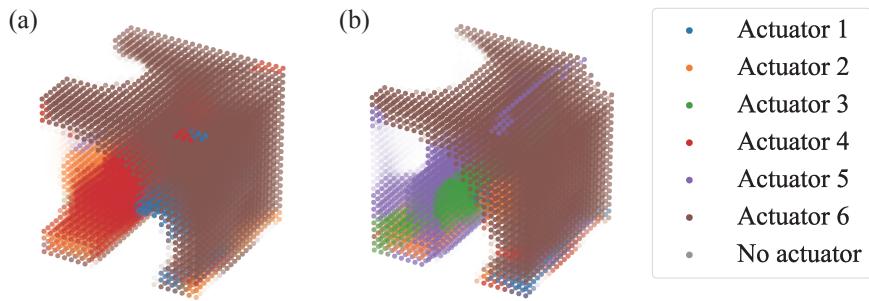


Figure 16: Optimized structures and actuator layouts (a) without the feedback controller and (b) with the feedback controller.

set the number of actuators as $N^{\text{act}} = 6$. We empirically set $w_\theta = 5$ for the weighting coefficient in the objective function (28). For terrain dataset, we assigned $\varsigma = 0.015$ m and $\varrho = 0.15$ m, respectively. We set $c^{\text{fb}} = 1/\varsigma = 200/3$ m $^{-1}$, the parameter for scaling the feedback signals in Eq. (25), which normalized the feedback signals independent of the terrain heights. We set the radius of the surrounding area $W = 0.3$ m, the target travel distance $L = 0.7$ m. We prepared 64($= N^{\text{data}}$) training data and also prepared 32 test data.

Figure 16 illustrates the optimized structures and actuation layouts obtained with and without the feedback controller. These structures resemble each other, that is, two leg-like substructures were formed and they were composed of different actuators.

Figure 17 illustrates the history of the objective function \mathcal{F} and constraint functions for binarization of material density and relaxed two-index variables, \mathcal{C}^{to} and \mathcal{C}^{lay} , during the optimization iterations where the objective function was plotted with a moving average taken every $N^{\text{data}}/N^{\text{batch}}$ iteration. For both cases, the objective function successfully decreased from 1.0 to around 0.5 with oscillation while constraint functions smoothly decreased to satisfy the constraints. These objective function values were larger than those of the two-dimensional problem, which suggests that the three-dimensional problem is more difficult than the two-dimensional one. This would be because terrains can also vary along with the depth direction and the soft robot requires to keep its posture more complexly. We would like to explore more sophisticated neural networks for the controller in our future work.

Figure 18 shows a violin plot illustrating the travel distances and objective function values evaluated

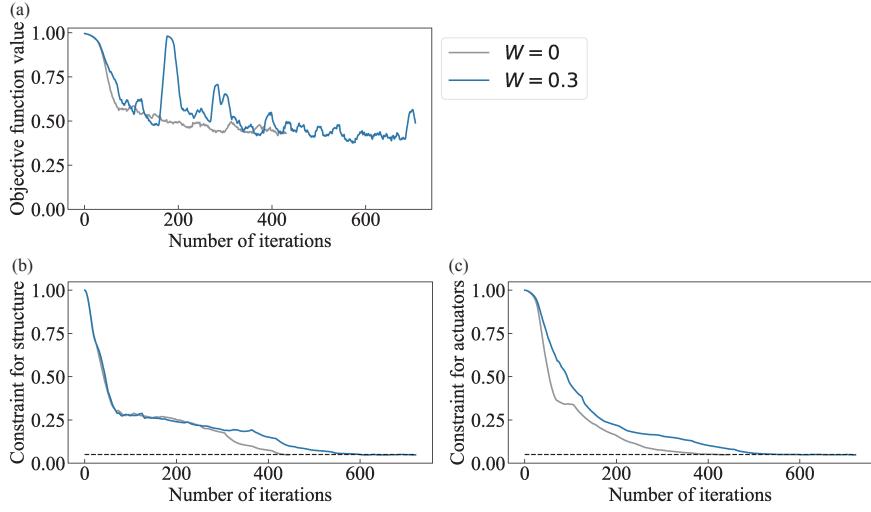


Figure 17: Trajectories of (a) objective function, (b) constraint function for binarization of structure, and (c) constraint function for binarization of actuator layout during optimization. The objective function values are plotted with a moving average taken every $N^{\text{data}}/N^{\text{batch}}$ iterations. Dashed lines in (b) and (c) represents the prescribed upper bounds of the constraint functions.

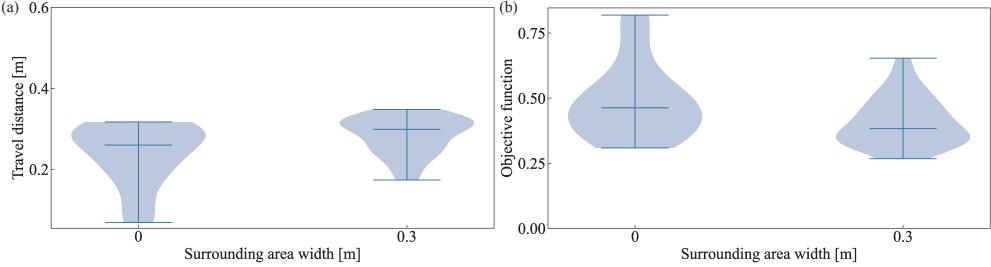


Figure 18: Violin plots illustrating (a) travel distances and (b) objective function values of the optimized soft robots for the test dataset.

using 32 test data for the soft robots with and without the feedback controller. While it is slight, the travel distance for $W = 0.3$ was larger on average than that for $W = 0$ and the objective function for $W = 0.3$ resulted in better performance on average than that for $W = 0$. The objective function values evaluated using the test dataset roughly range around 0.5, closely aligning with those obtained from the training dataset, which we again confirmed that the number of training data 64 is still enough for three-dimensional problem.

To analyze the differences in results with and without the feedback controllers in more detail, we provide the behaviors of optimized soft robots on two particular test terrains. Figure 19 shows the behaviors of optimized soft robots on the test terrain where the largest objective function was observed for the soft robot equipped with a feedforward controller ($W = 0$). The red and blue regions respectively represent areas undergoing expansion and contraction through actuation. The orange circle represents the range where the terrain feature is fed into the controller. In Fig. 19, the soft robot optimized by setting $W = 0$ got stuck in front of the uphill while that obtained by setting $W = 0.3$ advanced over the steps smoothly. Figure 20 illustrates actuation signals applied to the soft robot optimized by setting $W = 0.3$, that is, actuation signals applied to exhibit the dynamics of the right column of Fig. 19. Each line represents the deviation of each actuation signal from its mean value of all test data while each dashed line represents the actuation signal. The actuators 3 and 5 assigned to the rear *leg* and between *legs* exhibited the opposite tendency, and the deviation of actuator 3 is positive, which contributes to advancing over the uphill. Figure 21 shows the behaviors of optimized soft robots on the test terrain where the smallest objective function was observed

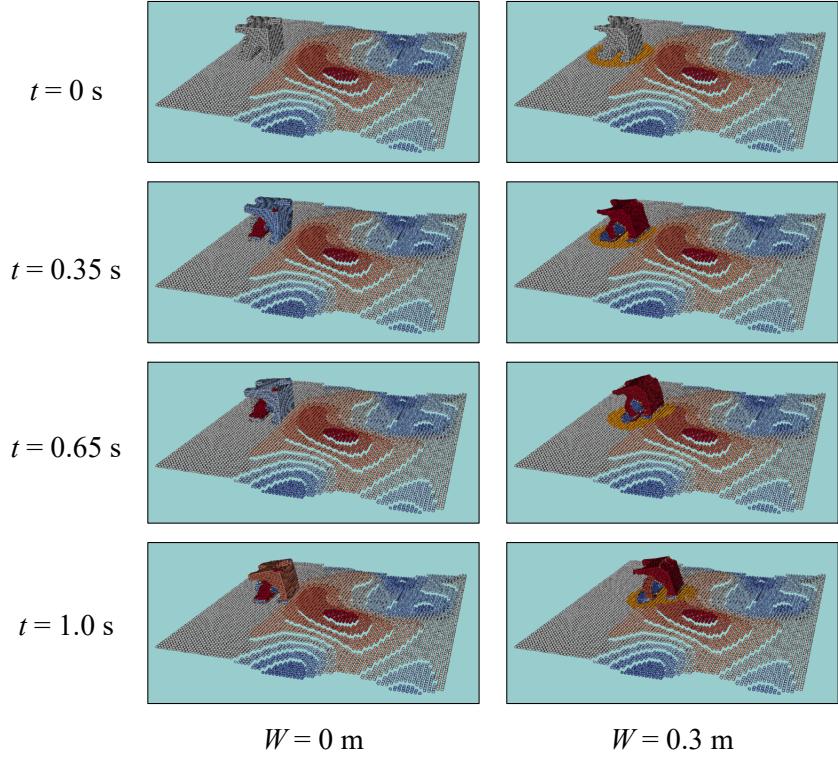


Figure 19: Dynamic behaviors of optimized soft robots for the test terrain on which the largest objective function was observed among the test dataset for the soft robot equipped with a feedforward controller ($W = 0$). The red and blue represent the expansion and contraction by actuation, respectively. The orange circle represents the range where the terrain height is inputted to the controller.

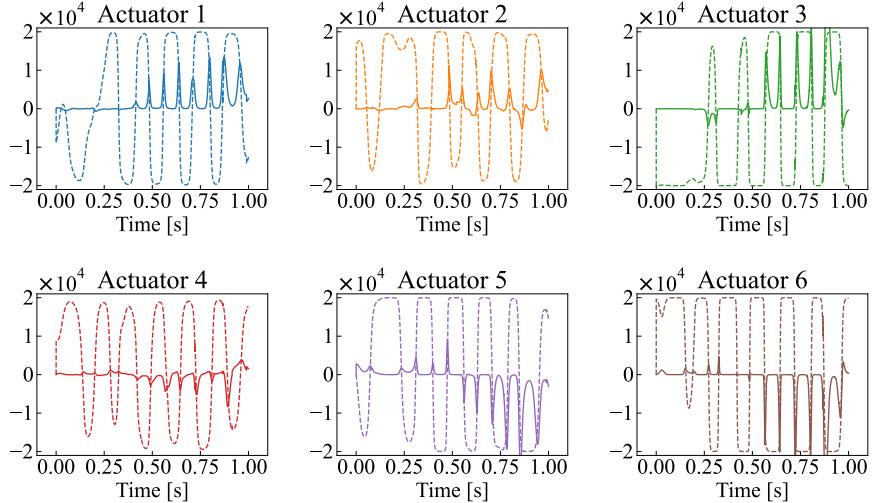


Figure 20: Signals of the respective actuators over the simulation duration. Each line represents the deviation of each actuation signal from its mean value of all test data. Each dashed line represents the actuation signal. Positive and negative actuation signals represent expansion and contraction, respectively.

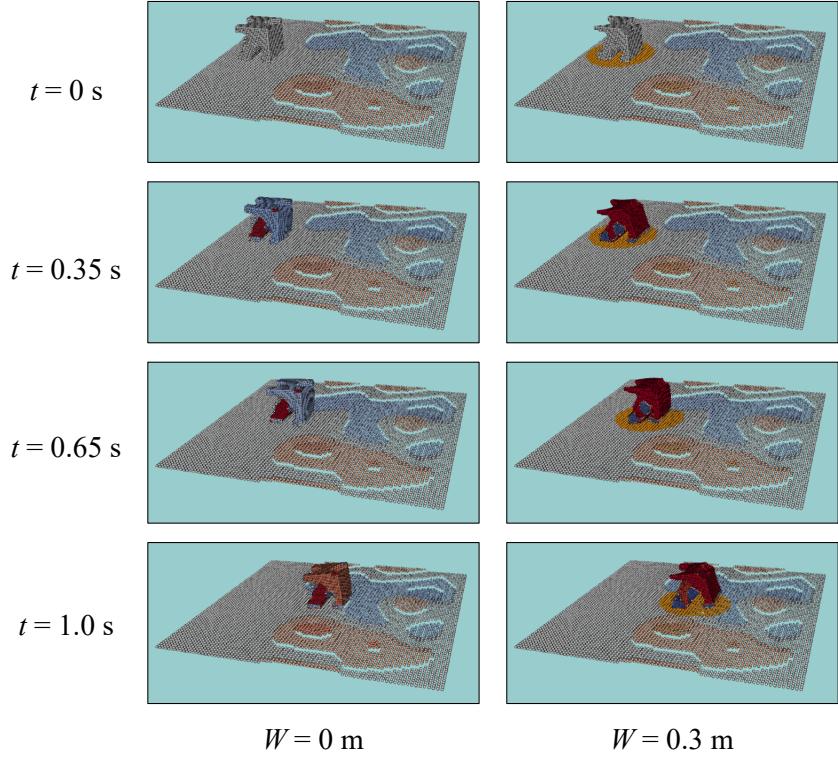


Figure 21: Dynamic behaviors of optimized soft robots for the test terrain on which the smallest objective function was observed among the test dataset for the soft robot equipped with a feedforward controller ($W = 0.3$). The red and blue represent the expansion and contraction by actuation, respectively. The orange circle represents the range where the terrain height is inputted to the controller.

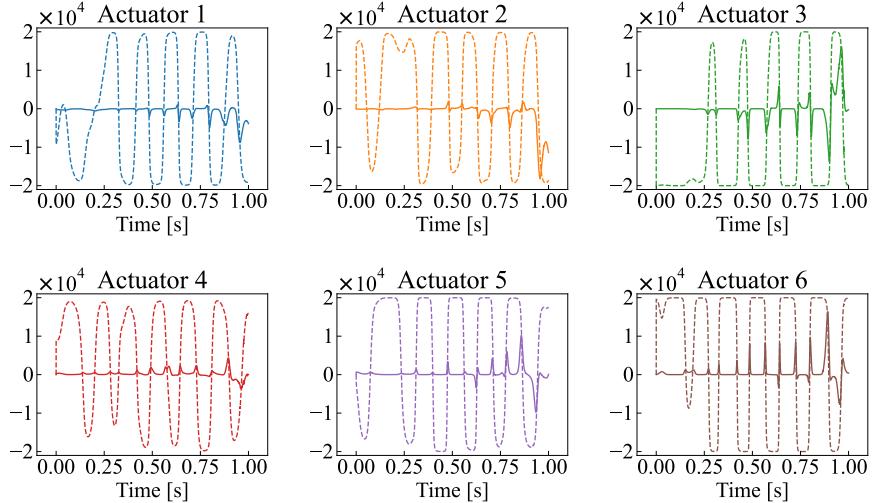


Figure 22: Signals of the respective actuators over the simulation duration. Each line represents the deviation of each actuation signal from its mean value of all test data. Each dashed line represents the actuation signal. Positive and negative actuation signals represent expansion and contraction, respectively.

for the soft robot equipped with a feedforward controller. In Fig. 21, the terrain in front of the soft robot is flat, so both the soft robots optimized with and without the controller advanced smoothly. Figure 22 shows actuation signals applied to exhibit the dynamics of the rightmost column of Fig. 21. We observe that the actuators 3 and 5 assigned to the rear *leg* and between *legs* exhibited the opposite tendency, which contributes to taking a large step forward, but there is no significant deviation because of the flatness of this terrain. These results again demonstrated the effectiveness of optimization of the feedback controller in addition to the structure.

5. Conclusions

This paper proposed the computational co-design of soft robots that can travel on varying terrains using the surrounding terrain features. We employed density-based topology optimization and relaxed multi-indexed optimization for structural and actuator layout optimizations, respectively. For controller designs, we employed a multilayer perceptron which output the actuation signal from the input of the terrain feature surrounding the soft robot. We formulated the simultaneous optimization problem of a structure and controller of a soft robot as the locomotion under the uncertainty in terrains and constructed an optimization algorithm, which relied on the stochastic gradient method. In numerical experiments, we obtained the optimized soft robots having animal-like structures consisting of *legs* and *tendons*, which are adaptively used by the trained controller to advance on various terrains. We also provided the post-forward analysis for optimized soft robots showcasing that the soft robots equipped with the feedback controller advanced more smoothly on various terrains.

In future work, we would like to integrate more state-of-the-art machine learning techniques, including convolution neural networks for processing environmental features and transformers for processing these time-series features, to handle various tasks beyond locomotion. We should also consider bridging the gap between simulation and real-world implementation. This involves the specific actuation model, such as pneumatic, electric, or magnetic actuators, and the manufacturing requirements, i.e., manufacturable structure and actuator layouts. We will address these issues in our future research and believe that the current study provides fundamental insight toward such a potential direction of co-designing soft robots.

References

- [1] C. Laschi, B. Mazzolai, M. Cianchetti, Soft robotics: Technologies and systems pushing the boundaries of robot abilities, *Science robotics* 1 (1) (2016) eaah3690.
- [2] C. Lee, M. Kim, Y. J. Kim, N. Hong, S. Ryu, H. J. Kim, S. Kim, Soft robot review, *International Journal of Control, Automation and Systems* 15 (2017) 3–15.
- [3] O. Yasa, Y. Toshimitsu, M. Y. Michelis, L. S. Jones, M. Filippi, T. Buchner, R. K. Katzschmann, An overview of soft robotics, *Annual Review of Control, Robotics, and Autonomous Systems* 6 (2023) 1–29.
- [4] N. Cheney, R. MacCurdy, J. Clune, H. Lipson, Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding, *ACM SIGEVolution* 7 (1) (2014) 11–23.
- [5] M. van Diepen, K. Shea, Co-design of the morphology and actuation of soft robots for locomotion, *Journal of Mechanical Design* 144 (8) (2022) 083305.
- [6] J. Bhatia, H. Jackson, Y. Tian, J. Xu, W. Matusik, Evolution gym: A large-scale benchmark for evolving soft robots, *Advances in Neural Information Processing Systems* 34 (2021) 2201–2214.
- [7] M. P. Bendsøe, N. Kikuchi, Generating optimal topologies in structural design using a homogenization method, *Computer methods in applied mechanics and engineering* 71 (2) (1988) 197–224.
- [8] O. C. Zienkiewicz, R. L. Taylor, *The finite element method: solid mechanics*, Vol. 2, Butterworth-heinemann, 2000.
- [9] T. J. Hughes, *The finite element method: linear static and dynamic finite element analysis*, Courier Corporation, 2012.
- [10] H. A. Eschenauer, N. Olhoff, Topology optimization of continuum structures: a review, *Appl. Mech. Rev.* 54 (4) (2001) 331–390.
- [11] J. D. Deaton, R. V. Grandhi, A survey of structural and multidisciplinary continuum topology optimization: post 2000, *Structural and Multidisciplinary Optimization* 49 (2014) 1–38.
- [12] T. Buhl, C. B. Pedersen, O. Sigmund, Stiffness design of geometrically nonlinear structures using topology optimization, *Structural and Multidisciplinary Optimization* 19 (2000) 93–104.
- [13] D. Jung, H. C. Gea, Topology optimization of nonlinear structures, *Finite Elements in Analysis and Design* 40 (11) (2004) 1417–1427.
- [14] X. S. Zhang, H. Chi, Z. Zhao, Topology optimization of hyperelastic structures with anisotropic fiber reinforcement under large deformations, *Computer Methods in Applied Mechanics and Engineering* 378 (2021) 113496.

- [15] S. M. Han, S. In Kim, Y. Y. Kim, Topology optimization of planar linkage mechanisms for path generation without prescribed timing, *Structural and Multidisciplinary Optimization* 56 (3) (2017) 501–517.
- [16] F. Fernandez, M. A. Puso, J. Solberg, D. A. Tortorelli, Topology optimization of multiple deformable bodies in contact with large deformations, *Computer Methods in Applied Mechanics and Engineering* 371 (2020) 113288.
- [17] A. H. Frederiksen, O. Sigmund, K. Poulios, Topology optimization of self-contacting structures, *Computational Mechanics*.
- [18] A. Stomakhin, C. Schroeder, L. Chai, J. Teran, A. Selle, A material point method for snow simulation, *ACM Transactions on Graphics (TOG)* 32 (4) (2013) 1–10.
- [19] A. Stomakhin, C. Schroeder, C. Jiang, L. Chai, J. Teran, A. Selle, Augmented mpm for phase-change and varied materials, *ACM Transactions on Graphics (TOG)* 33 (4) (2014) 1–11.
- [20] C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, A. Selle, The material point method for simulating continuum materials, in: *ACM SIGGRAPH 2016 Courses*, 2016, pp. 1–52.
- [21] Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana, C. Jiang, A moving least squares material point method with displacement discontinuity and two-way rigid body coupling, *ACM Transactions on Graphics* 37 (4) (2018) 150.
- [22] Y. Li, X. Li, M. Li, Y. Zhu, B. Zhu, C. Jiang, Lagrangian–eulerian multidensity topology optimization with the material point method, *International Journal for Numerical Methods in Engineering* 122 (14) (2021) 3400–3424.
- [23] Y. Sato, H. Kobayashi, C. Yuhn, A. Kawamoto, T. Nomura, N. Kikuchi, Topology optimization of locomoting soft bodies using material point method, *Structural and Multidisciplinary Optimization* 66 (3) (2023) 50.
- [24] C. Yuhn, Y. Sato, H. Kobayashi, A. Kawamoto, T. Nomura, 4D topology optimization: Integrated optimization of the structure and self-actuation of soft bodies for dynamic motions, *Computer Methods in Applied Mechanics and Engineering* 414 (2023) 116187.
- [25] H. Kobayashi, F. Gholami, S. M. Montgomery, M. Tanaka, L. Yue, C. Yuhn, Y. Sato, A. Kawamoto, H. J. Qi, T. Nomura, Computational synthesis of locomotive soft robots by topology optimization, *Science Advances* (in press).
- [26] T. George Thuruthel, Y. Ansari, E. Falotico, C. Laschi, Control strategies for soft robotic manipulators: A survey, *Soft robotics* 5 (2) (2018) 149–163.
- [27] K. Chin, T. Hellebrekers, C. Majidi, Machine learning for soft robotic sensing and control, *Advanced Intelligent Systems* 2 (6) (2020) 1900171.
- [28] D. Kim, S.-H. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee, et al., Review of machine learning methods in soft robotics, *Plos one* 16 (2) (2021) e0246102.
- [29] R. V. Woldseth, N. Aage, J. A. Barentzen, O. Sigmund, On the use of artificial neural networks in topology optimisation, *Structural and Multidisciplinary Optimization* 65 (10) (2022) 294.
- [30] D. W. Abueidda, S. Koric, N. A. Sobh, Topology optimization of 2d structures with nonlinearities using deep learning, *Computers & Structures* 237 (2020) 106283.
- [31] S. Zheng, Z. He, H. Liu, Generating three-dimensional structural topologies via a u-net convolutional neural network, *Thin-Walled Structures* 159 (2021) 107263.
- [32] L. Zheng, K. Karapiperis, S. Kumar, D. M. Kochmann, Unifying the design space and optimizing linear and nonlinear truss metamaterials by generative modeling, *Nature Communications* 14 (1) (2023) 7563.
- [33] Y. Jiang, C. Yang, J. Na, G. Li, Y. Li, J. Zhong, A brief review of neural networks based learning and control and their applications for robots, *Complexity* 2017 (1) (2017) 1895897.
- [34] T. Du, K. Wu, P. Ma, S. Wah, A. Spielberg, D. Rus, W. Matusik, Diffpd: Differentiable projective dynamics, *ACM Transactions on Graphics (TOG)* 41 (2) (2021) 1–21.
- [35] M. P. Bendsøe, Optimal shape design as a material distribution problem, *Structural optimization* 1 (1989) 193–202.
- [36] M. P. Bendsoe, O. Sigmund, *Topology optimization: theory, methods, and applications*, Springer Science & Business Media, 2003.
- [37] M. P. Bendsøe, O. Sigmund, Material interpolation schemes in topology optimization, *Archive of applied mechanics* 69 (1999) 635–654.
- [38] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*, Academic press, 2014.
- [39] D. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, in: *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.
- [40] Y. Hu, T.-M. Li, L. Anderson, J. Ragan-Kelley, F. Durand, Taichi: a language for high-performance computation on spatially sparse data structures, *ACM Transactions on Graphics (TOG)* 38 (6) (2019) 201.
- [41] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, F. Durand, DiffTaichi: differentiable programming for physical simulation, *ICLR*.
- [42] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: *Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings*, 2010, pp. 249–256.