Supplementary Information

Reinforcement learning in linear embedding space unlocks generalizable control across soft robot configurations

Supplementary information contains:

Supplementary Section 1. Koopman embedding training
Supplementary Section 2. Feedforward policy calculation
Supplementary Section 3. Policy learning with warm-start
Supplementary Section 4. Pseudocode of LERL
Supplementary Section 5. Action space re-allocation
Supplementary Section 6. Policy adaptation to rigid-body robots
Supplementary Section 7. Feedback policy initialization
Supplementary Section 8. Theoretical results
Supplementary Section 8. Design parameters of honeycomb segments
Supplementary Figure. Soft robotic platforms for experimental validation
Supplementary Figure. Auxiliary policy adaptation performance in target reaching tasks
Supplementary Figure. Auxiliary policy adaptation performance in bow tie tracking tasks
Supplementary Figure. Fast-moving traction stick tracking performance
Supplementary Table. Computational time per step on the Type I elephant-trunk robots
Supplementary Table. Hyperparameters
Supplementary Table. Comparison of three types of segments

Supplementary Section 1. Koopman embedding training

The 20k collected samples are used to train the Koopman embedding on a laptop with an NVIDIA GeForce RTX 4090 GPU. Hyperparameters for offline Koopman embedding training are provided in Supplementary Table 1.

The regression loss L_e is the mean squared error between the lifted actual and predicted states over p steps, which measures the prediction accuracy in the embedding space, i.e.,

$$L_e = \frac{1}{p}\sum_{i=1}^{p} \|\mathbf{s}_{k+i} - \hat{\mathbf{s}}_{k+i}\|^2, \tag{S1}$$

where $\hat{\mathbf{s}}_{k+i} = \mathbf{A}^i \mathbf{s}_k + \sum_{j=1}^{i} \mathbf{A}^{j-1}\mathbf{B}\mathbf{u}_{k+i-j}$ is the i-th step ahead state prediction in the linear embedding space from time instant k.

The reconstruction loss L_r is the mean squared error between the actual states and their predicted reconstructed counterparts over p steps, that is,

$$L_r = \frac{1}{p}\sum_{i=1}^{p} \|\mathbf{x}_{k+i} - \hat{\mathbf{x}}_{k+i}\|^2, \tag{S2}$$

where the $\hat{\mathbf{x}}_{k+i} = \mathbf{T}^{-1}\mathbf{s}_{k+i}$ is the prediction of the reconstruction state.

To guarantee the matrix $\mathbf{T}$ in equation (S2) to be invertible, we parameterize $\mathbf{T}$ as a trainable tensor and introduce the following regularization loss:

$$L_T = \sum_{i=1}^{12} \|\lambda(\mathbf{T})_i - 1\|^2, \tag{S3}$$

to shape its eigenvalues near the unit circle.

Supplementary Section 2. Feedforward policy calculation

A feedforward policy $\mathbf{u}_r$ is incorporated into the control strategy (equation (6) in Methods) to maintain the robot tip in the desired pose when the dynamic feedback policy $\mathbf{u}_e = \mathbf{K}\mathbf{s}_e$ vanishes. However, this static feedforward policy is not directly available, as the robot tip's static model is unknown. We address this challenge by learning the feedforward policy offline using precollected quasi-static motion data[1]. The embedding network, an MLP with an architecture of 12-32-64-32-m and Rectified Linear Unit (ReLU) activation functions, is adopted to generate $\mathbf{u}_r$. Here, m denotes the input dimension of the current robot configuration, determined by the robot type and the number of segments. The Adam optimizer is adopted to train the network parameters offline, using $N_f = 500$ quasi-static motion samples of $\{\mathbf{u}_r, \mathbf{x}_r\}$ (collected by the following sinusoidal excitation (S5)) to minimize the following mean squared error:

$$E_r = \frac{1}{N_f}\sum_{i=1}^{N_f} \|\mathbf{u}_{r,i} - \mathbf{\Phi}(\mathbf{x}_{r,i})\|^2. \tag{S4}$$

The resulting feedforward policy for each robot configuration is represented as $\mathbf{u}_r = \mathbf{\Phi}(\mathbf{x}_r)$, where $\mathbf{\Phi}(\cdot)$ is the pretrained MLP tailored for each robot configuration.

Data collection for feedforward policy training. The 500 quasi-static motion samples of $\{\mathbf{u}_r, \mathbf{x}_r\}$ were collected with the following sinusoidal input excitation:

$$\begin{aligned}
&\mathbf{u}_r = [u_{4r-3}, u_{4r-2}, u_{4r-1}, u_{4r}]^\top, \\
&u_{4r-3} = D_{4r-3}(\sin(\omega_{4r-3}\pi t + \phi_{4r-3}) + 0.99a), \\
&u_{4r-2} = D_{4r-2}(\sin(\omega_{4r-2}\pi t + \phi_{4r-2}) + 0.99b), \\
&u_{4r-1} = D_{4r-1}(\sin(\omega_{4r-1}\pi t + \phi_{4r-1}) + 0.99c), \\
&u_{4r} = D_{4r}(\sin(\omega_{4r}\pi t + \phi_{4r}) + 0.99d), \\
&D_i \in \frac{1}{20}[1, 2, \ldots, 10]^\top, \;\; \omega_i \sim \mathrm{Unif}(0.1, 0.4), \;\; i = 4r-3, 4r-2, 4r-1, 4r, \\
&\phi_{4r-3} \sim \mathrm{Unif}(-1, 1), \;\; \phi_{4r-2} \sim \mathrm{Unif}(-1, 1) \;\; \phi_{4r-1} \sim \mathrm{Unif}(-2, 2) \;\; \phi_{4r} \sim \mathrm{Unif}(-2, 2), \\
&a, b \sim \mathrm{Unif}(0, 0.5), \;\; c, d \sim \mathrm{Unif}(0.5, 1).
\end{aligned} \tag{S5}$$

where $\mathbf{u}_r \in \mathbb{R}^4$ denotes the control input for r-th segment; $r = 1, 2, 3, \cdots$ represents the number of the segment; $\mathrm{Unif}(l_1, l_2)$ denotes a uniform distribution from l_1 to l_2; t denotes the time index.

We additionally collected offline feedforward training data using the following random input excitation, and compared the performance with that using sinusoidal input excitation (Fig. 8b). The associated random input excitation was designed as follows:

$$\begin{aligned}
&\mathbf{u}_r = [u_{4r-3}, u_{4r-2}, u_{4r-1}, u_{4r}]^\top, \\
&\mathbf{u}_i = D_i D_t, \;\; i = 4r-3, 4r-2, 4r-1, 4r, \\
&D_i \in \frac{1}{20}[1, 2, \ldots, 10]^\top, \;\; D_t \sim \mathrm{Unif}(0, 1).
\end{aligned} \tag{S6}$$

Supplementary Section 3. Policy learning with warm-start

Here, we present the implementation of policy learning with warm-start to further enhance learning efficiency. Even in scenarios with nonlinear system dynamics, a stabilizing control policy can still be effectively derived. In this work, a straightforward way is to carry out a model-based control design using offline learned Koopman models, which makes full use of the baseline control policy $\mathbf{u}_{e,b}$ and the associated $\mathbf{H}_b$ matrix.

Let $\Delta\mathbf{H}_i = \mathbf{H}_b - \hat{\mathbf{H}}_i$, equation (10) in Methods is equivalent to solving:

$$\min_{\Delta\mathbf{h}_{i+1}} \|\Delta\mathbf{h}_{i+1}^\top \mathbf{z}_k - \Delta\mathbf{d}(\mathbf{z}_k, \Delta\mathbf{H}_i)\|^2, \tag{S7}$$

where $\Delta\mathbf{d}(\mathbf{z}_k, \Delta\mathbf{H}_i) = \mathbf{H}_b^\top \mathbf{z}_k - \mathbf{d}(\mathbf{z}_k, \hat{\mathbf{H}}_i)$.

Then, we derive the update for $\Delta\mathbf{h}_{i+1}$ as

$$\Delta\mathbf{h}_{i+1} = (\bar{\mathbf{Z}}\bar{\mathbf{Z}}^\top)^{-1}\bar{\mathbf{Z}}\Delta\mathbf{W}, \tag{S8}$$

where $\Delta\mathbf{W} = [\Delta\mathbf{d}(\mathbf{z}_{[1]}, \Delta\mathbf{h}_i) \cdots \Delta\mathbf{d}(\mathbf{z}_{[N]}, \Delta\mathbf{h}_i)]^\top$.

Remark 1 *Note that the baseline control policy $\mathbf{u}_{e,b}$ is not necessarily computed from a model-based controller design. The associated matrix $\mathbf{H}_b$ could be estimated by recursively solving equation (10) with an available control policy $\mathbf{u}_{e,b}$.*

Let a lower bound of the computational residual error of $\mathbf{h}_i$ be $\epsilon_{h,i}$. The following proposition states the advantage of equation (S8) over equation (11) in numerical precision.

Proposition 1 (Numerical precision error) *If $\mathbf{H}_b$ is such that $\Delta\mathbf{H}_i \succ 0$ and there exists $\delta > 1$ with $\delta\Delta\mathbf{d} \leq d$, then the numerical precision error of $\mathbf{h}_i$ with the operation in equation (S8) is smaller than $\delta^{-1}\epsilon_{h,i}$.*

Proof. Note that the numerical computational error is due to the inverse operation and the weak fulfillment of $\mathbf{Z}\mathbf{Z}^\top$ being full rank. Let the error in operation $(\mathbf{Z}\mathbf{Z}^\top)^{-1}$ associated with the i-th iteration be $\epsilon_{z,i}$. Then, we have $\epsilon_{h,i+1} = \epsilon_{z,i}\mathbf{Z}\mathbf{W}$. Similarly, the numerical error of $\Delta\mathbf{h}_{i+1}$ is $\epsilon_{\Delta\mathbf{h},i+1} = \epsilon_{z,i}\mathbf{Z}\Delta\mathbf{W}$. In view of the fact $\delta\Delta\mathbf{d} \leq d$ and the definition of $\mathbf{W}$ and $\Delta\mathbf{W}$, we have $\epsilon_{\Delta\mathbf{h},i+1} \leq \epsilon_{z,i}\mathbf{Z}\delta^{-1}\mathbf{W} = \delta^{-1}\epsilon_{h,i}$. Note that $\hat{\mathbf{H}}_i = \Delta\mathbf{H}_i + \mathbf{H}_b$, hence the computation of the vectorization of $\hat{\mathbf{H}}_i$ using the operation in equation (S8) results in a numerical precision error smaller than $\delta^{-1}\epsilon_{h,i}$. □

Supplementary Section 4. Pseudocode of LERL

The main implementation steps of linear embedding RL (LERL) for policy learning across diverse configurations are outlined in Algorithm 1. These steps include offline Koopman embedding pretraining and online policy learning. The offline phase establishes baseline capabilities, while the online phase allows adaptive refinement for improved performance. The online update of $\mathbf{h}$ can follow three strategies: (a) least-square (LS) rule: update $\mathbf{h}$ using equation (11); (b) LS rule with warm-start: update $\Delta\mathbf{h}$ using equation (S8) and set $\mathbf{h} = \mathbf{h}_b - \Delta\mathbf{h}$; (c) gradient-descent rule: update $\mathbf{h}$ using the Adam optimizer.

Supplementary Algorithm 1 Pseudocode of LERL for multiple configurations

Offline design on single honeycomb segment:

Collect open-loop input-state data; // use sinusoidal input excitations.
Construct $\mathbf{\Psi}(\mathbf{x})$ with equation (2); // MLP with a 12-64-128-64-12 network structure.
Offline train the Koopman embedding with the loss in equation (5);
Offline train the feedforward policy using equation (S4); // MLP with a 12-32-64-32-m network structure.

Offline design for new configurations:

Collect 500 input-state samples in the quasi-static condition; // use sinusoidal input excitations.
Offline train the feedforward policy for each new configuration using equation (S4);

Online policy learning for diverse configurations:

Require: $i = 0$, $N_{\max}$, $\{\mathbf{x}_r\}$.

Determine the input dimension m of the current robot configuration;
Initialize $\mathbf{u}_e$ and $\hat{\mathbf{H}}$ with those from the transferred policy;
while $i \leq N_{\max}$ **do**
    Measure state $\mathbf{x}_i$, generate $\mathbf{s}_r$ and $\mathbf{s}_i$ using the pretrained Koopman embedding;
    Generate the feedforward policy $\mathbf{u}_r = \mathbf{\Phi}(\mathbf{s}_r)$ for the current configuration;
    Calculate $\mathbf{u}_{e,i} = -\hat{\mathbf{H}}_{uu,i}^{-1}\hat{\mathbf{H}}_{us,i}\mathbf{s}_{e,i} + \mathbf{n}_{e,i}$, in which $\mathbf{n}_{e,i}$ is applied in initial several steps;
    Apply $\mathbf{u}_i = \mathbf{u}_{e,i} + \mathbf{u}_r$ to the robot and calculate $d(\mathbf{z}_i, \hat{\mathbf{H}}_i) = l(\mathbf{z}_i) + \gamma\|\mathbf{z}_i^+\|_{\mathbf{H}_i}^2$;
    Update $\mathbf{h}_{i+1}$; // Choose strategy (c) (Adam optimizer) with limited data, or strategies (a) and (b) (LS rule and LS rule with warm-start) when sufficient data is available.
    Construct $\hat{\mathbf{H}}_{i+1}$ with $\mathbf{h}_{i+1}$;
    Set $i = i + 1$;
end while

Supplementary Section 5. Action space re-allocation

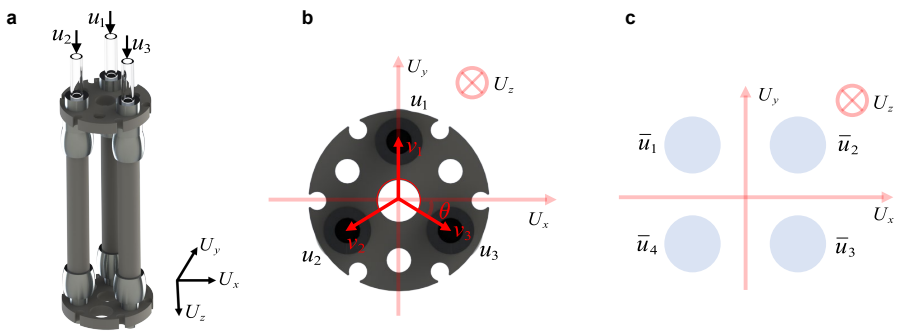
Policy transfer to elephant-trunk soft robots. We adopt the padding procedure to transfer the feedback policy $\mathbf{u}_e = \mathbf{K}\mathbf{s}_e = \mathbf{K}(\mathbf{\Psi}(x) - \Psi(x_d))$, $\mathbf{K} \in \mathbb{R}^{4\times 24}$, trained on the honeycomb segment $\mathrm{E}_{\mathrm{S}}^{1}$, to elephant-trunk soft robots in multiple configurations. For a configuration with n segments, we transfer the policy gain $\mathbf{K}$ by replicating it n times, forming $\mathbf{K}_{n,0} = [\mathbf{K}; \cdots; \mathbf{K}] \in \mathbb{R}^{4n\times 24}$. The policy gain $\mathbf{K}_{n,0}$ then initializes the policy learning process for the new configuration and is fine-tuned online.

Policy transfer to soft-muscle robots. Each elephant-trunk segment has four control inputs, while the soft-muscle segment has three. To allow direct policy transfer from the elephant-trunk robot to the soft-muscle robot, we use an action re-allocation strategy. In particular, we first represent the transferred policy from the elephant-trunk robot as a virtual control policy $\bar{\mathbf{u}} \in \mathbb{R}^4$ for the soft-muscle robot, which is then mapped to three channels using:

$$\mathbf{u} = \mathbf{T}_a \cdot \bar{\mathbf{u}}, \tag{S9}$$

where the action distribution matrix $\mathbf{T}_a \in \mathbb{R}^{3\times 4}$ is derived from equation (S10) to equation (S12).

The above re-allocation strategy allows directly transferring the control policy from the elephant-trunk robot to the soft-muscle and hybrid-assembled robots, without any prior modelling procedure.



Supplementary Fig. 1: **Mechanism of the action re-allocation strategy. a**, Illustration of the equivalent actuation $\mathbf{U}$ of a soft-muscle segment. **b**, Illustration of the intermediate variable $\mathbf{v}$, $\theta = \pi/6$. **c**, Illustration of the virtual control $\bar{\mathbf{u}}$.

Calculation of matrix $\mathbf{T}_a$. The soft-muscle segment has two bending degrees of freedom (DoF) and one elongation DoF, represented by the introduced equivalent actuation $\mathbf{U} = [U_x, U_y, U_z]^\top$ (Supplementary Fig. 1a). Using the virtual variable $\mathbf{v} = [v_1, v_2, v_3]^\top$ as a bridge (Supplementary Fig. 1b), the following matrix $\mathbf{T}_{a_1}$ is computed to map the real actuation $\mathbf{u} = [u_1, u_2, u_3]^\top$ to the equivalent actuation $\mathbf{U}$, i.e.,

$$\begin{bmatrix} U_x \\ U_y \\ U_z \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 3\sqrt{3}/4 & -3\sqrt{3}/4 \\ -3/2 & 3/4 & 3/4 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}}_{\mathbf{T}_{a_1}} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}. \tag{S10}$$

The virtual actuation $\bar{\mathbf{u}} = [\bar{u}_1, \bar{u}_2, \bar{u}_3, \bar{u}_4]^\top$ (Supplementary Fig. 1c) is mapped to the equivalent actuation $\mathbf{U}$ through a matrix $\mathbf{T}_{a_2}$, i.e.,

$$\begin{bmatrix} U_x \\ U_y \\ U_z \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}}_{\mathbf{T}_{a_2}} \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \\ \bar{u}_3 \\ \bar{u}_4 \end{bmatrix}. \tag{S11}$$

Combing equation (S10) with equation (S11), one has

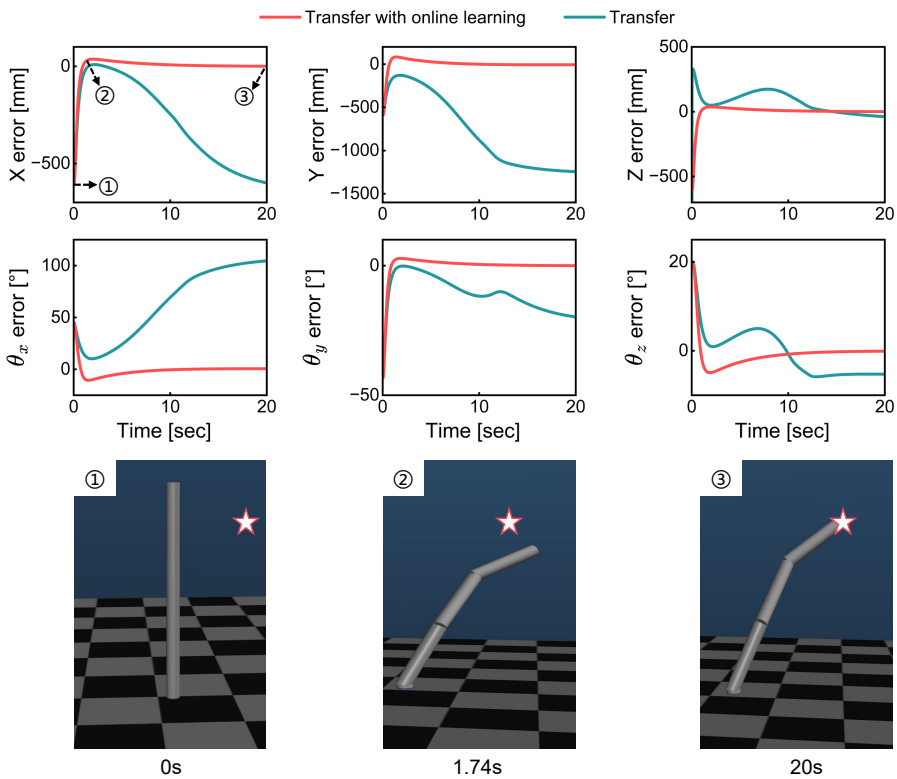
$$\mathbf{T}_a = \mathbf{T}_{a_1}^{-1} \mathbf{T}_{a_2}. \tag{S12}$$

Supplementary Section 6. Policy adaptation to rigid-body robots

To further validate the versatility of our LERL framework, we transfer the policy learned on the soft robotic segment E_S^1 to a three-segment rigid-body robot for a target reaching task. The rigid-body robot is designed in the Mujoco environment (Supplementary Fig. 2), with each segment actuated by the virtual force $\mathbf{u}_{\text{rig}} = [u_x, u_y]^\top \in \mathbb{R}^2$ on the plane vertical to the rigid body. The controlled states, i.e., the robot tip's pose and velocity, remain consistent with those in the soft robot scenario. We use the action allocation strategy introduced in Supplementary Section 5 to transfer the policy $\mathbf{u} \in \mathbb{R}^4$ learned on the honeycomb segment to each segment of the rigid-body robot, i.e.,

$$\mathbf{u}_{\text{rig}} = \begin{bmatrix} -1/2 & -1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & -1/2 & 1/2 \end{bmatrix} \mathbf{u}. \tag{S13}$$

As shown in Supplementary Fig. 2, directly transferring the policy via the control action re-allocation strategy in equation (S13) leads to instability in the target-reaching task. This is because the virtual forces in the rigid-body robot differ in physical interpretation from the air voltage inputs in the honeycomb segment. However, online policy learning via model-free RL in the shared Koopman embedding space, pretrained on the honeycomb segment E_S^1, enables rapid convergence to the target. This demonstrates the effectiveness of our LERL framework in fast adaptation to rigid-body robots. Future work will explore its applicability across various real-world rigid-body robotic systems.



Supplementary Fig. 2: **Policy adaptation to rigid robots.** Pose error curves for policy transfer with or without online learning scenarios, along with the pose of the rigid-body robot at selected moments. The highlighted star is the reference target of the robot tip.

Supplementary Section 7. Feedback policy initialization

The elephant-trunk robot and the worm robot utilize different actuation modes: positive and negative pressure actuation, respectively. Thereby, a significant gap arises between these heterogeneous robots within the same pretrained Koopman embedding when transferring the policy trained on the honeycomb segment to the worm robots.

To bridge the aforementioned gap, we effortlessly initialize the feedback policy $\mathbf{u}_e$ using just 10k samples $\{\mathbf{x}, \mathbf{u}\}$ via KCID. Specifically, given the pretrained Koopman embedding $\Psi(\cdot)$ from the honeycomb segment, the model parameters $\mathbf{A}$ and $\mathbf{B}$ are first calculated via solving:

$$\min_{\mathbf{A},\mathbf{B}} \sum_{i=1}^{N_0} \|\mathbf{s}_{i+1} - \mathbf{A}\mathbf{s}_i - \mathbf{B}u_i\|^2, \tag{S14}$$

where $\mathbf{s}_{i+1} = \Psi(\mathbf{x}_{i+1})$ and $\mathbf{s}_i = \Psi(\mathbf{x}_i)$, $N_0 = 10k$.

Using the calculated $\mathbf{A}$ and $\mathbf{B}$, an initialized feedback policy $\mathbf{u}_e = \mathbf{K}_0\mathbf{s}_e$ is generated by solving the standard LQR problem:

$$\begin{aligned} &\min_{\mathbf{u}_e} \sum_{k=0}^{+\infty} \mathbf{s}_{e,k}^\top \mathbf{Q}_0 \mathbf{s}_{e,k} + \mathbf{u}_{e,k}^\top \mathbf{R}_0 \mathbf{u}_{e,k}, \\ &\text{s.t. } \mathbf{s}_{e,k+1} = \mathbf{A}\mathbf{s}_{e,k} + \mathbf{B}\mathbf{u}_{e,k}, \end{aligned} \tag{S15}$$

where $\mathbf{Q}_0 \in \mathbb{R}^{n_\psi \times n_\psi}$ and $\mathbf{R}_0^{m\times m}$ are chosen as positive-definite matrices.

Supplementary Section 8. Theoretical results

This section theoretically demonstrates that, under mild assumptions, LERL with the least-square update rule (11) outperforms a robust KCID that accounts for modelling errors.

Assume that the reference pair $\{\mathbf{x}_r, \mathbf{u}_r\}$ is reachable, i.e., $\mathbf{x}_r = \mathbf{f}(\mathbf{x}_r, \mathbf{u}_r)$, where $\mathbf{f}(\cdot,\cdot)$ is the unknown dynamics of the soft robot. The dynamic model of the state error is described as $\mathbf{x}_e^+ = \mathbf{f}_e(\mathbf{x}_e, \mathbf{u}_e) := \mathbf{f}(\mathbf{x}, \mathbf{u}) - \mathbf{f}(\mathbf{x}_r, \mathbf{u}_r)$. The resulting error dynamics on the embedding space is given as

$$\mathbf{s}_e^+ = \mathbf{A}\mathbf{s}_e + \mathbf{B}\mathbf{u}_e + \mathbf{w}_e(\mathbf{s}_e, \mathbf{u}_e). \tag{S16}$$

According to optimal control theory, a stabilizing feedback policy $\mathbf{u}_e = \mathbf{K}\mathbf{s}_e$ results in a quadratic value function $V(\mathbf{s}_e) = \mathbf{s}_e^\top P \mathbf{s}_e$ when $\mathbf{w}_e(\mathbf{s}_e, \mathbf{u}_e) = 0$ holds, where $\mathbf{P} \in \mathbb{R}^{n_\psi \times n_\psi}$ is a symmetric positive definite matrix. However, since $\mathbf{w}_e(\mathbf{s}_e, \mathbf{u}_e)$ is non-negligible, the feedback policy $\mathbf{u}_e$, applied to the model in equation (1), leads to a time-dependent value function $V(\mathbf{s}_{e,k}) = \mathbf{s}_{e,k}^\top \mathbf{P}_k \mathbf{s}_{e,k}$, where $\mathbf{P}_k$ varies with time. Hence, the so-called Q-function (state-action value function), i.e., $Q(\mathbf{z}_k) := l(\mathbf{z}_k) + \gamma V(\mathbf{A}\mathbf{s}_{e,k} + \mathbf{B}\mathbf{u}_{e,k} + \mathbf{w}_{e,k})$, can be rewritten as

$$Q(\mathbf{z}_k) = \mathbf{z}_k^\top \mathbf{H}_o \mathbf{z}_k + G(\mathbf{z}_k) := \mathbf{z}_k^\top \mathbf{H}(\mathbf{P}_k) \mathbf{z}_k, \tag{S17}$$

where $\mathbf{z}_k := [\mathbf{s}_{e,k}^\top,\ \mathbf{u}_{e,k}^\top]^\top$; $\mathbf{H}_o = \begin{bmatrix} \mathbf{Q} + \gamma \mathbf{A}^\top \mathbf{P}_k \mathbf{A} & \gamma \mathbf{A}^\top \mathbf{P}_k \mathbf{B} \\ \gamma \mathbf{B}^\top \mathbf{P}_k \mathbf{A} & \mathbf{R} + \gamma \mathbf{B}^\top \mathbf{P}_k \mathbf{B} \end{bmatrix}$; $\mathbf{H}(\mathbf{P}_k) = \begin{bmatrix} \mathbf{H}_{ss}(\mathbf{P}_k) & \mathbf{H}_{su}(\mathbf{P}_k) \\ \mathbf{H}_{us}(\mathbf{P}_k) & \mathbf{H}_{uu}(\mathbf{P}_k) \end{bmatrix}$ is a matrix on $\mathbf{P}_k$, expressed in a block form similar to $\mathbf{H}_o$; the term $G(\mathbf{z}_k)$ originates from the modelling error. Since it is directly tied to the input-state pair, we leverage $G(\mathbf{z}_k)$ to enhance control performance, rather than treating it as an uncertainty that degrades performance.

To prove the theoretical property, we introduce the following assumptions about the equivalent Koopman error model in equation (S16).

Lemma 1 (Equivalent Koopman error model[2])
(a) Model (S16) is stabilizable at $\mathbf{s}_e = 0$ if and only if $\mathbf{f}_e(\mathbf{x}_e, \mathbf{u}_e)$ is stabilizable.
(b) The model uncertainty $\mathbf{w}_e(\mathbf{s}_e, \mathbf{u}_e) = \Psi(\mathbf{f}(\Psi^{-1}(\mathbf{s}_e), \mathbf{u}_e)) - \mathbf{A}\mathbf{s}_e - \mathbf{B}\mathbf{u}_e$ satisfies:

$$\|\mathbf{w}_e(\mathbf{s}_e, \mathbf{u}_e)\|^2 \le L_s \|\mathbf{s}_e\|^2 + L_u \|\mathbf{u}_e\|^2, \tag{S18}$$

where L_s and L_u are bounded constants.

Assumption 1 *There exists an $\epsilon > 0$ such that the following Lyapunov equation has a solution on $\bar{\mathbf{P}} \in \mathbb{R}^{n_\psi \times n_\psi}$ and $\bar{\mathbf{K}} \in \mathbb{R}^{m \times n_\psi}$, i.e.,*

$$\bar{\mathbf{P}} = \tilde{\mathbf{Q}} + \bar{\mathbf{K}}^\top \tilde{\mathbf{R}} \bar{\mathbf{K}} + \gamma \mathbf{F}^\top \tilde{\mathbf{P}} \mathbf{F}, \tag{S19}$$

where $\tilde{\mathbf{P}} = \bar{\mathbf{P}} + \epsilon \bar{\mathbf{P}}^2$, $\tilde{\mathbf{Q}} = \mathbf{Q} + \gamma(1+\epsilon)/\epsilon L_s$, $\tilde{\mathbf{R}} = \mathbf{R} + \gamma(1+\epsilon)/\epsilon L_u$.

Remark 2 *An optimal solution can be calculated by setting $\partial \bar{\mathbf{P}} / \partial \mathbf{K} = 0$, which leads to:*

$$\bar{\mathbf{K}} = -\gamma \mathbf{T}(\bar{\mathbf{P}})^{-1} \mathbf{B}^\top \tilde{\mathbf{P}} \mathbf{A}, \tag{S20}$$

where $\mathbf{T}(\mathbf{P}) = \tilde{\mathbf{R}} + \gamma \mathbf{B}^\top \tilde{\mathbf{P}} \mathbf{B}$. Substituting equation (S20) to the right-hand side of equation (S19), the result follows:

$$\bar{\mathbf{P}} = \tilde{\mathbf{Q}} - \gamma^2 \mathbf{A}^\top \tilde{\mathbf{P}} \mathbf{B} \mathbf{T}(\mathbf{P})^{-1} \mathbf{B}^\top \tilde{\mathbf{P}} \mathbf{A} + \gamma \mathbf{A}^\top \tilde{\mathbf{P}} \mathbf{A}, \tag{S21}$$

which is the variant of the so-called Riccati equation.

Proposition 2 *The Hessian matrix of $G(\mathbf{z}_k)$, i.e., $\Delta \mathbf{G}(\mathbf{z}_k)$, is bounded as*

$$\Delta \mathbf{G}(\mathbf{z}_k) \le \mathbf{Y}(\mathbf{P}_k), \tag{S22}$$

where

$$\mathbf{Y}(\mathbf{P}_k) = \begin{bmatrix} \gamma\frac{1+\epsilon}{\epsilon}L_s + \epsilon\gamma\mathbf{A}^\top\mathbf{P}_k^2\mathbf{A} & \gamma\epsilon\mathbf{A}^\top\mathbf{P}_k^2\mathbf{B} \\ \gamma\epsilon\mathbf{B}^\top\mathbf{P}_k^2\mathbf{A} & \gamma(\frac{1}{\epsilon}+1)L_u + \gamma\epsilon\mathbf{B}^\top\mathbf{P}_k^2\mathbf{B} \end{bmatrix}.$$

Moreover, if s spans an invariant Koopman subspace for the dynamic model of soft robots, then $G(z) = 0$.

***Proof**. Note that one can rewrite equation (8) as*

$$\begin{aligned} Q(\mathbf{z}_k) &= \mathbf{s}_{e,k}^\top\mathbf{Q}\mathbf{s}_{e,k} + \mathbf{s}_{e,k}^\top\mathbf{K}^\top\mathbf{R}\mathbf{K}\mathbf{s}_{e,k} + \gamma(\mathbf{F}\mathbf{s}_{e,k} + \mathbf{w}_{e,k})^\top\mathbf{P}_k(\mathbf{F}\mathbf{s}_{e,k} + \mathbf{w}_{e,k}) \\ &\leq \mathbf{s}_{e,k}^\top\mathbf{Q}\mathbf{s}_{e,k} + \mathbf{s}_{e,k}^\top\mathbf{K}^\top\mathbf{R}\mathbf{K}\mathbf{s}_{e,k} + \gamma\mathbf{s}_{e,k}^\top\mathbf{F}^\top(\mathbf{P}_k + \epsilon\mathbf{P}_k^2)\mathbf{F}\mathbf{s}_{e,k} + \gamma\frac{1+\epsilon}{\epsilon}\mathbf{s}_{e,k}^\top\mathbf{L}\mathbf{s}_{e,k}, \end{aligned} \tag{S23}$$

where the last inequality is due to the following fact: Given any constant $\epsilon > 0$, it holds that

$$2\mathbf{s}_{e,k}^\top\mathbf{F}^\top\mathbf{P}\mathbf{w}_{e,k} \leq \epsilon\mathbf{s}_{e,k}^\top\mathbf{F}^\top\mathbf{P}^2\mathbf{F}\mathbf{s}_{e,k} + \frac{1}{\epsilon}\mathbf{s}_{e,k}^\top\mathbf{L}\mathbf{s}_{e,k},$$

where $\mathbf{L} = L_s + L_u\mathbf{K}^\top\mathbf{K}$. In view of equation (S23), one has:

$$G(\mathbf{z}_k) \leq \mathbf{z}_k^\top\mathbf{Y}_k\mathbf{z}_k,$$

i.e., the condition in equation (S22) holds. Moreover, if s spans an invariant Koopman subspace for the dynamic model for soft robots, the uncertainty in equation (1) holds $\mathbf{w} = \mathbf{0}$ and it leads to $\Delta\mathbf{G}(z) = 0$.

To state the following Theorems 1 and 2 in a compact form, let $\bar{\mathbf{P}}^*$ and $\bar{\mathbf{K}}^*$ be the optimal solution of equation (S19) and denote $\bar{\mathbf{H}}_i(\bar{\mathbf{P}}_i) = \mathbf{\Xi}_i(\bar{\mathbf{P}}_i) + \mathbf{Y}_i(\bar{\mathbf{P}}_i)$, where $\bar{\mathbf{P}}_i$ is calculated using a value iteration procedure as follows with $\bar{\mathbf{P}}_0 = 0$.

1. Value update:

$$\bar{\mathbf{P}}_{i+1} = \tilde{\mathbf{Q}} - \gamma^2\mathbf{A}^\top\tilde{\mathbf{P}}_i\mathbf{B}\mathbf{T}(\bar{\mathbf{P}}_i)^{-1}\mathbf{B}^\top\tilde{\mathbf{P}}_i\mathbf{A} + \gamma\mathbf{A}^\top\tilde{\mathbf{P}}_i\mathbf{A}. \tag{S24a}$$

2. Policy update:

$$\bar{\mathbf{K}}_{i+1} = -\gamma\mathbf{T}(\bar{\mathbf{P}}_i)^{-1}\mathbf{B}^\top\tilde{\mathbf{P}}_i\mathbf{A}. \tag{S24b}$$

Theorem 1 *Under Assumption 1 and $\bar{\mathbf{Z}}\bar{\mathbf{Z}}^\top$ is full rank, given $\hat{\mathbf{H}}_0 = \bar{\mathbf{H}}_0(\mathbf{P}_0)$ and $\mathbf{K}_0 = \bar{\mathbf{K}}_0$, the iteration of $\hat{\mathbf{H}}_i$, using the least-square update rule (11), satisfies $\hat{\mathbf{H}}_i \leq \bar{\mathbf{H}}_i(\bar{\mathbf{P}}_i)$ for any $i \in \mathbb{N}$. Moreover, $\hat{\mathbf{H}}_\infty \leq \bar{\mathbf{H}}(\bar{\mathbf{P}}^*) := \bar{\mathbf{H}}^*$.*

Proof. One first rewrites:

$$\begin{aligned} d(\mathbf{z}_{i,k}, \hat{\mathbf{H}}_i) &= l(\mathbf{z}_{i,k}) + \mathbf{z}_{i,k+1}^\top\gamma\hat{\mathbf{H}}_i\mathbf{z}_{i,k+1} \\ &\leq \mathbf{z}_k^\top\mathrm{vec}\left(\mathbf{\Xi}_i(\mathbf{P}_{k,i}) + \mathbf{Y}_i(\mathbf{P}_{k,i})\right), \end{aligned} \tag{S25}$$

where $\mathbf{P}_i = \begin{bmatrix}\mathbf{I} & \mathbf{K}_i^\top\end{bmatrix}\hat{\mathbf{H}}_i\begin{bmatrix}\mathbf{I} & \mathbf{K}_i^\top\end{bmatrix}^\top$. Note that the computation of $\bar{\mathbf{H}}_i(\bar{\mathbf{P}}_i)$ via the iteration on $\bar{\mathbf{P}}_i$ with equation (S24a) is equivalent to the iteration on $\mathbf{H}_{i+1}$ under $\mathbf{H}_0 = \bar{\mathbf{H}}_0(\bar{\mathbf{P}}_0)$.

Hence, by using the Kronecker products, from equation (11) and equation (S25), one computes:

$$\begin{aligned} \mathbf{h}_{i+1} &\leq (\bar{\mathbf{Z}}\bar{\mathbf{Z}}^\top)^{-1}(\bar{\mathbf{Z}}\bar{\mathbf{Z}}^\top)\mathrm{vec}\left(\mathbf{\Xi}_i(\mathbf{P}_{k,i}) + \mathbf{Y}_i(\mathbf{P}_{k,i})\right) \\ &= \mathrm{vec}\left(\mathbf{\Xi}_i(\mathbf{P}_{k,i}) + \mathbf{Y}_i(\mathbf{P}_{k,i})\right) \\ &\leq \mathrm{vec}\left(\mathbf{\Xi}_i(\bar{\mathbf{P}}_i) + \mathbf{Y}_i(\bar{\mathbf{P}}_i)\right), \end{aligned} \tag{S26}$$

where $\mathbf{h} = \mathrm{vec}(\hat{\mathbf{H}})$, the last equality holds since $\hat{\mathbf{H}}_0 = \bar{\mathbf{H}}_0$ and $\mathbf{K}_0 = \bar{\mathbf{K}}_0$.

In view of the last inequality in equation (S26), it holds that:

$$\hat{\mathbf{H}}_{i+1} \leq \bar{\mathbf{H}}_{i+1}(\bar{\mathbf{P}}_{i+1}). \tag{S27}$$

Similar to the proof argument[3], the convergence property of the value iteration procedure as in equation (S24) can be stated, that is:

i)$\bar{\mathbf{P}}_1 \leq \bar{\mathbf{P}}_2 \leq \bar{\mathbf{P}}^*$;
ii) $\lim_{i\to+\infty} \bar{\mathbf{P}}_i = \bar{\mathbf{P}}^*$, $\lim_{i\to+\infty} \bar{\mathbf{K}}_i = \bar{\mathbf{K}}^*$.
Hence, $\hat{\mathbf{H}}_\infty \leq \bar{\mathbf{H}}_\infty(\bar{\mathbf{P}}_\infty) = \hat{\mathbf{H}}_\infty(\bar{\mathbf{P}}_\infty) + Y_\infty(\bar{\mathbf{P}}_\infty) = \bar{\mathbf{H}}^*$.

Remark 3 *Equation (S27) in Theorem 1 shows that the overall cost-to-go with the least-square update rule (11) is smaller than that using the robust value iteration with the procedure (S24). The latter results in a robust LQR as iteration i goes to infinity. Hence, LERL with the least-square update rule (11) outperforms the robust KCID with the procedure (S24).*

Theorem 2 *Let $\bar{\mathbf{s}}_{e,k} = \gamma^{k/2}\mathbf{s}_{e,k}$, then the control input computed by:*

$$\bar{\mathbf{u}}_{e,k} = -(\hat{\mathbf{H}}_{uu,\infty})^{-1}\hat{\mathbf{H}}_{us,\infty}\bar{\mathbf{s}}_{e,k}, \tag{S28}$$

stabilizes $\bar{\mathbf{s}}_{e,k}$.

Poof. Consider two Lyapunov functions:

$$\bar{L}(\mathbf{s}_{e,k}) = \|\mathbf{s}_{e,k}\|^2_{\bar{\mathbf{P}}^*}, \tag{S29a}$$

under $\bar{\mathbf{u}}_{e,k} = \bar{\mathbf{K}}\bar{\mathbf{s}}_{e,k}$, and

$$L(\mathbf{s}_{e,k}) = \|(\mathbf{s}_{e,k}, \bar{\mathbf{u}}_{e,k})\|^2_{\hat{\mathbf{H}}_\infty}, \tag{S29b}$$

under control equation (S28). We first show the convergence under the Lyapunov function $\bar{L}(\mathbf{s}_{e,k})$ with $\bar{\mathbf{u}} = \bar{\mathbf{K}}\bar{\mathbf{s}}_{e,k}$. To this end, one writes:

$$L(\mathbf{s}_{e,k+1}) - L(\mathbf{s}_{e,k}) = \|\mathbf{s}_{e,k+1}\|^2_{\mathbf{P}} - \|\mathbf{s}_{e,k}\|^2_{\mathbf{P}}. \tag{S30}$$

Consider $\bar{\mathbf{s}}_{e,k} = \gamma^{k/2}\mathbf{s}_{e,k}$. In view of equation (1), one can write:

$$\begin{aligned} \bar{\mathbf{s}}_{e,k+1} &= \gamma^{1/2}(\mathbf{A}\bar{\mathbf{s}}_{e,k} + \mathbf{B}\bar{\mathbf{u}} + \bar{\mathbf{w}}_{e,k}) \\ &= \gamma^{1/2}(\mathbf{F}\bar{\mathbf{s}}_{e,k} + \bar{\mathbf{w}}_{e,k}), \end{aligned} \tag{S31}$$

where the second equality is due to the definition $\bar{\mathbf{u}}_{e,k} = \bar{\mathbf{K}}^*\bar{\mathbf{s}}_{e,k}$. Taking equation (S31) into equation (S30) leads to:

$$\begin{aligned} L(\bar{\mathbf{s}}_{e,k+1}) - L(\bar{\mathbf{s}}_{e,k}) &= \gamma\|\mathbf{F}\bar{\mathbf{s}}_{e,k} + \bar{\mathbf{w}}_{e,k}\|^2_{\mathbf{P}} - \|\bar{\mathbf{s}}_{e,k}\|^2_{\mathbf{P}} \\ &\leq \|\bar{\mathbf{s}}_{e,k}\|^2_{\gamma(1+\epsilon)/\epsilon L + \gamma\mathbf{F}^\top(\mathbf{P}+\epsilon\mathbf{P}^2)\mathbf{F} - \mathbf{P}} \\ &= -\|\bar{\mathbf{s}}_{e,k}\|^2_{\bar{\mathbf{Q}}+\mathbf{K}^\top\mathbf{R}\mathbf{K}} \\ &< 0. \end{aligned} \tag{S32}$$

Hence, the state $\bar{\mathbf{s}}_{e,k}$ converges to the origin asymptotically under $\bar{\mathbf{u}}_{e,k} = \bar{\mathbf{K}}^*\bar{\mathbf{s}}_{e,k}$.

In view of the results that $\hat{\mathbf{H}}_\infty \leq \bar{\mathbf{H}}^*$ and also note that $\bar{L}(\mathbf{s}_{e,k}) = \|\mathbf{z}_k\|^2_{\bar{\mathbf{H}}^*}$, we get:

$$L(\mathbf{s}_{e,k}) \leq \bar{L}(\mathbf{s}_{e,k}),\ \forall k \in \mathbb{N}.$$

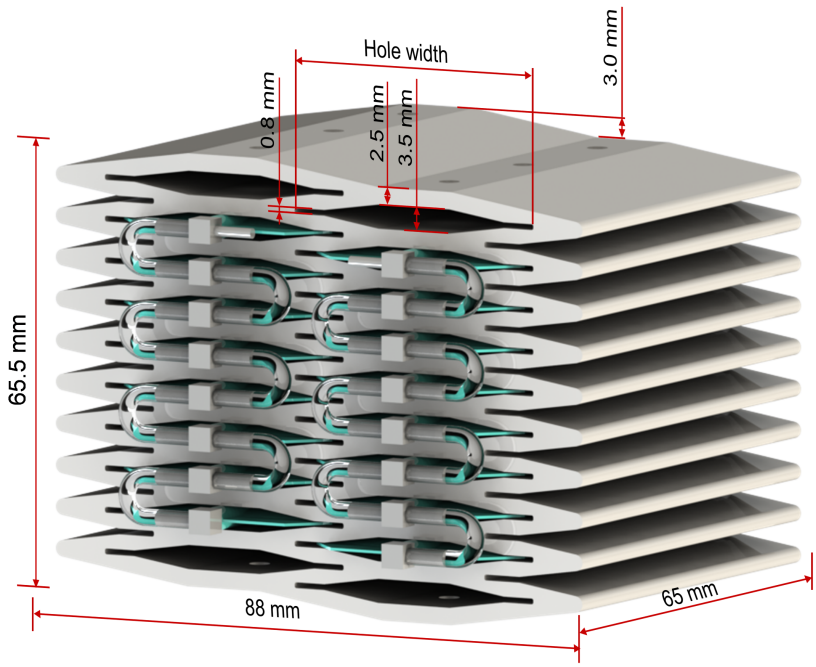
Remark 4 *Theorem 2 implies that under the control policy in equation (S28), the state $\mathbf{s}_{e,k}$ converges to the origin asymptotically when γ is close to 1.*

Supplementary Section 9. Design parameters of honeycomb segments

The Type I honeycomb segment is designed across five stiffness levels (Supplementary Table 1). The stiffness level of each segment is determined by the honeycomb hole width shown in Supplementary Fig. 3. Specifically, the superscript "1" denotes the lowest stiffness level, corresponding to a honeycomb hole size of 45 mm. The stiffness level increases with the superscript index when the honeycomb hole size decreases, as summarized in Supplementary Table 1.

Supplementary Table 1. Relationship between stiffness levels and hole widths of the honeycomb segment.

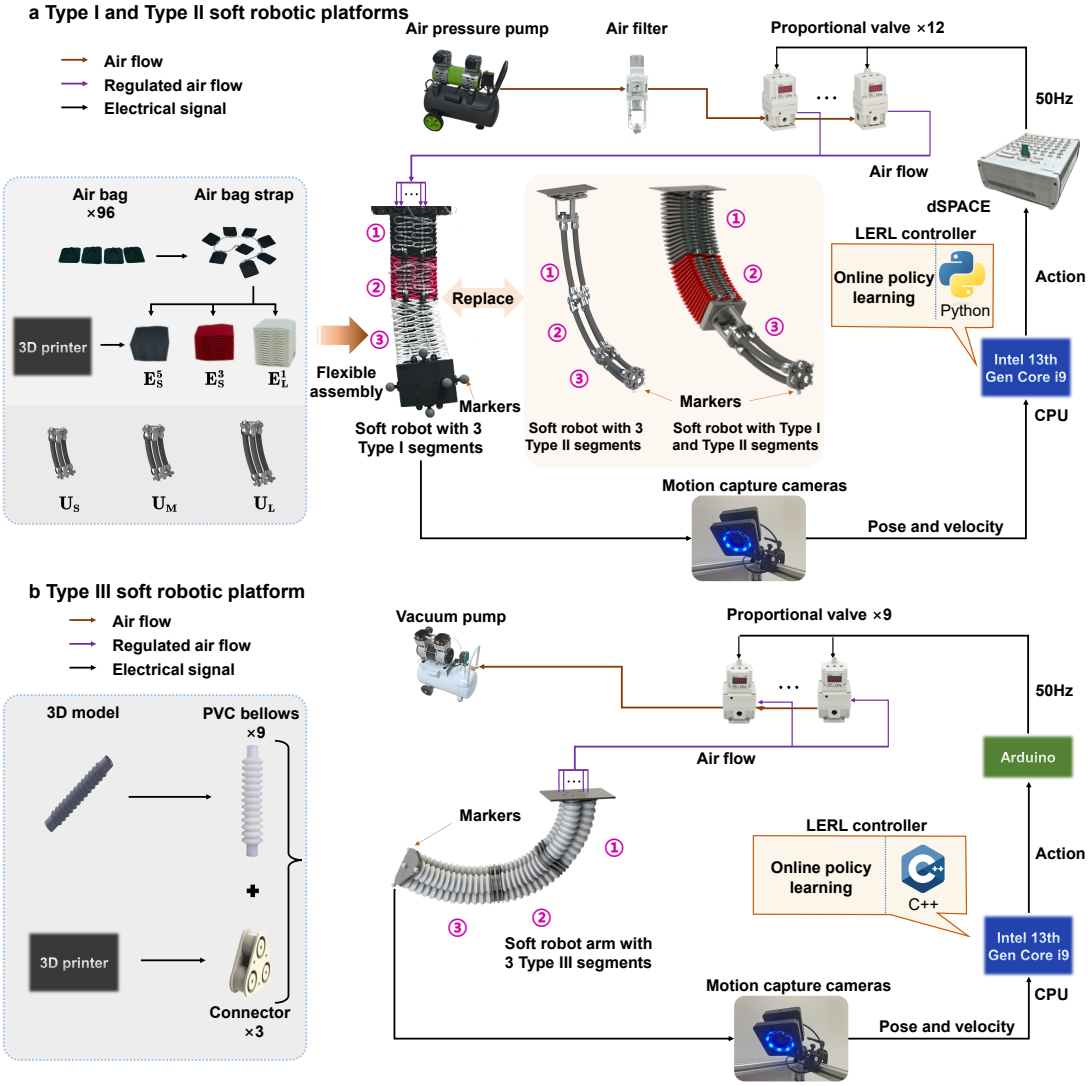
Stiffness level	**1**	**2**	**3**	**4**	**5**
Hole width	45 mm	43 mm	41 mm	38 mm	35 mm



Supplementary Fig. 3: **Schematic diagram of the honeycomb segment structure.** The stiffness level of the honeycomb segment can be adjusted by modifying the hole width.

Supplementary Figure. Soft robotic platforms for experimental validation

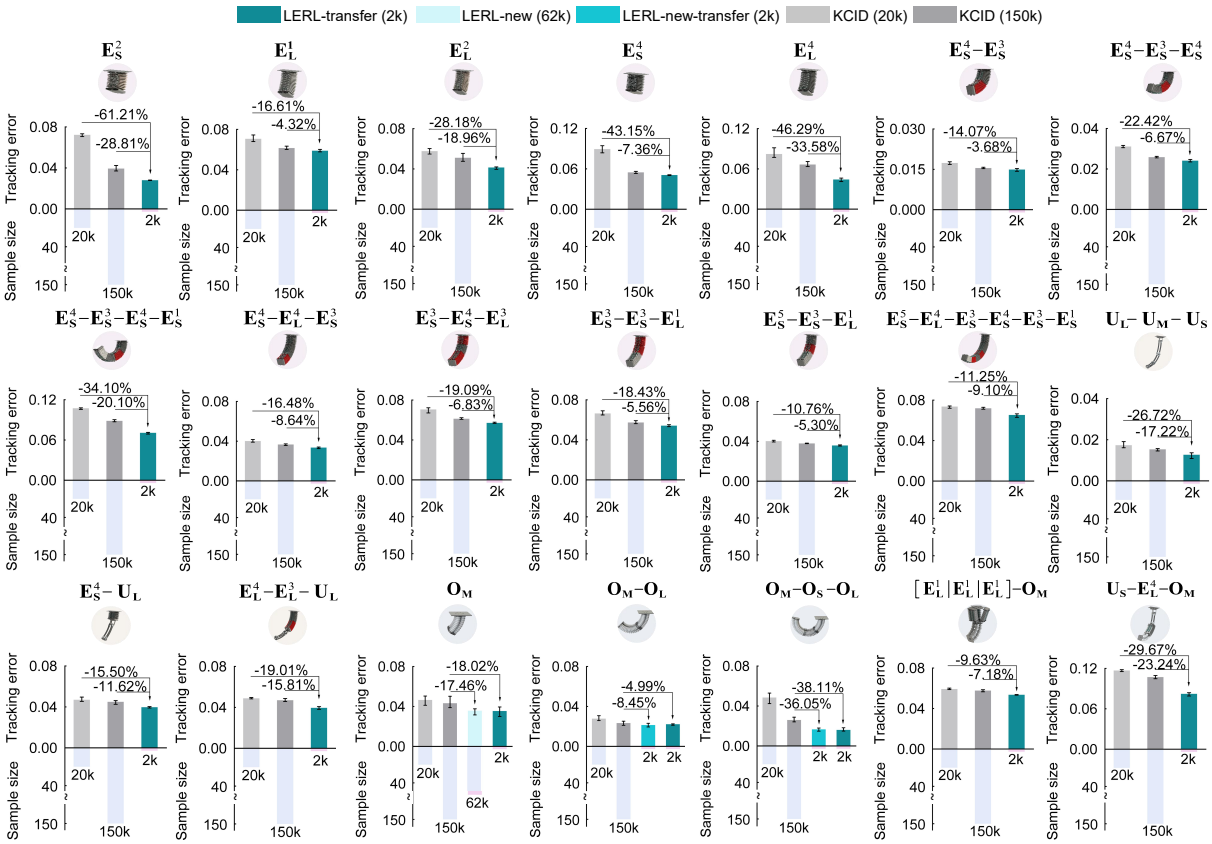
Three types of soft robot platforms with diverse configurations are used for experimental validation. Type I and Type II soft robots (Supplementary Fig. 4a) share the same motion capture module and the computing unit, but with different pneumatic actuation systems across different configurations. Note that, in principle, a Type I robot with six segments requires 24 proportional valves (i.e., 24 control channels), since each elephant-trunk segment is actuated by four valves. However, the Type I platform provides only 16 D/A channels. To accommodate this hardware constraint, the control inputs are shared between adjacent segments: segments 1–2, 3–4, and 5–6, each pair shares the same control inputs. Type III soft robotic platform (Supplementary Fig. 4b) utilizes the same motion capture module but implements the controller in C++ on a CPU (Intel Core i9-13900HX). Three soft robotic platforms, with their multiple configurations (including hybrid-assembled configurations), can perform various tasks.



Supplementary Fig. 4: **Types I-III soft robot platforms. a**, Type I and Type II soft robotic platforms in specific configurations. **b**, Type III soft robotic platform in a particular setup.

Supplementary Figure. Auxiliary policy adaptation performance in target reaching tasks

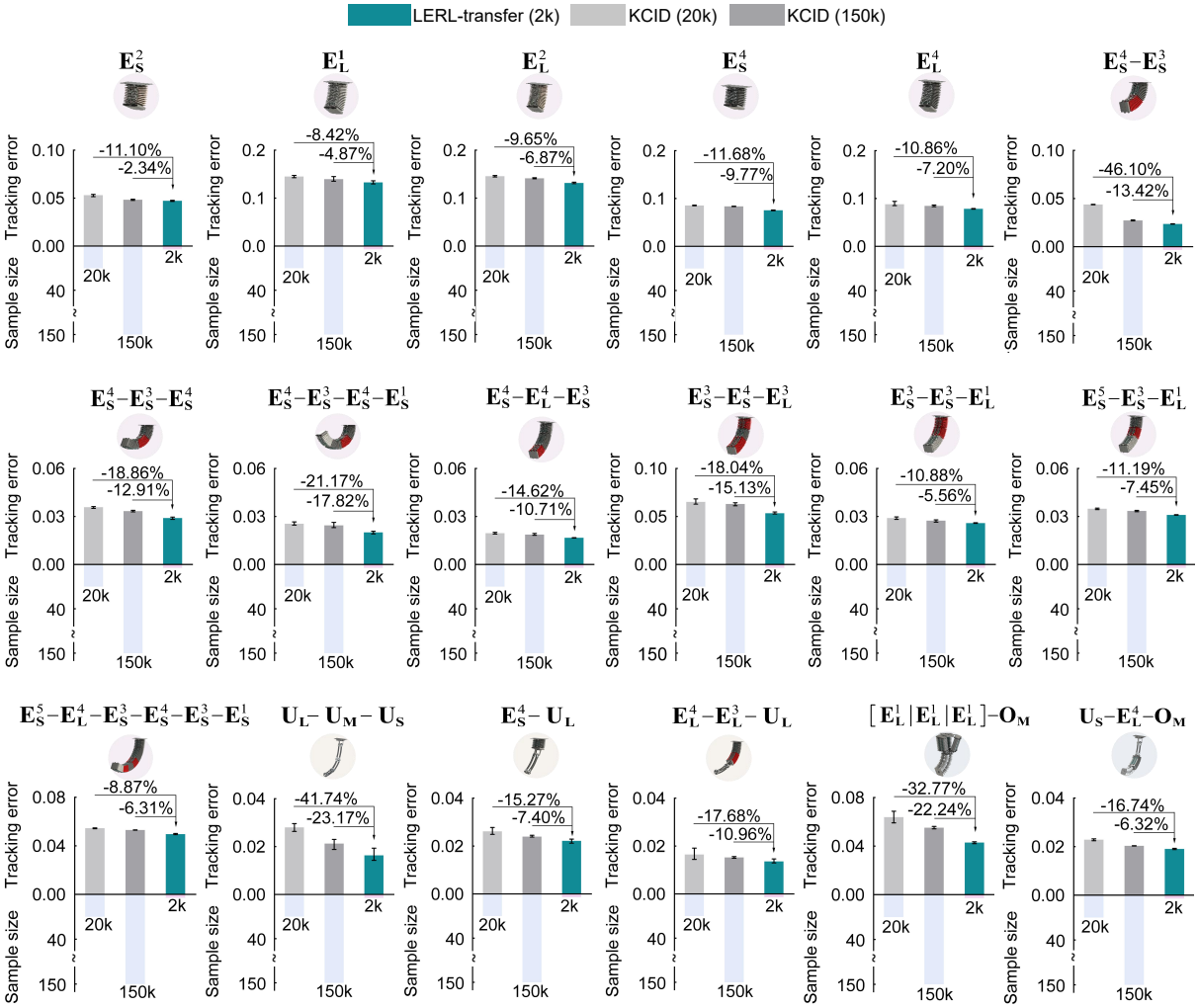
The policy adaptation performance in the target reacking task across 21 configurations demonstrates that LERL outperforms KCID in both tracking error and sample size reduction (Supplementary Fig. 5).



Supplementary Fig. 5: **Auxiliary policy adaptation performance across diverse configurations in target reaching tasks.** Target reaching performance comparisons across 21 configurations from three soft robot prototypes, wherein Tracking error $:= \frac{1}{M \cdot N} \sum_{i=1}^{M} \sum_{k=1}^{N} \left\| \bar{\mathbf{x}}_{i,k} - \bar{\mathbf{x}}_{r,i,k} \right\|^2$, $\bar{\mathbf{x}}_{i,k}, \bar{\mathbf{x}}_{r,i,k} \in \mathbb{R}^{12}$ represent the normalized real state variable (pose and velocity) and its reference at the k-th step for the i-th experiment, N denotes the algorithm horizon, and $M = 5$ is the number of repeated experiments. The policy adaptation to worm robots' configurations uses policy learning with a newly trained Koopman embedding (LERL-new). The Koopman embedding for the single worm segment $\mathrm{O_M}$ is trained with 20k samples, and the policy is updated online from scratch using 2k samples. The policy is then transferred and refined online (LERL-new-transfer) to different configurations with 2k samples under the newly trained Koopman embedding. Each error bar represents the full range of the tracking error, from the minimum to the maximum value.

Supplementary Figure. Auxiliary policy adaptation performance in bow tie tracking tasks

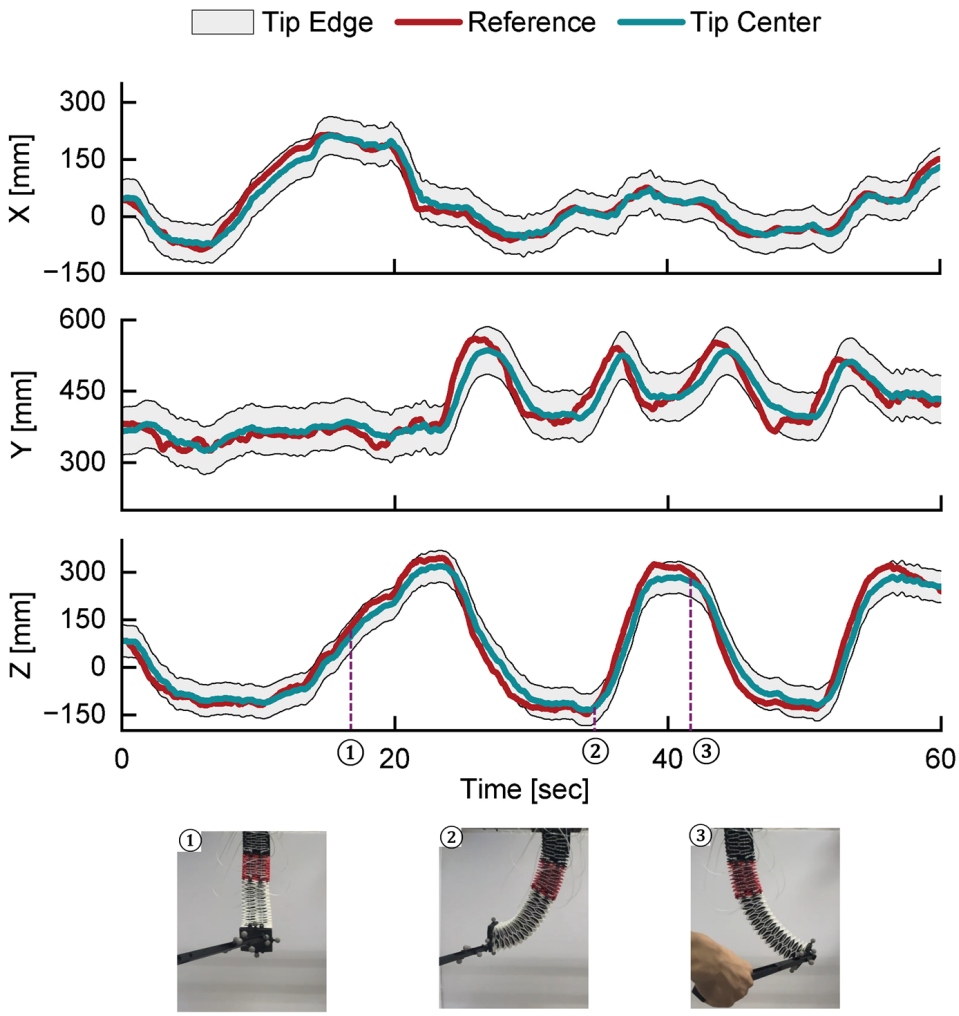
The policy adaptation performance in the bow tie tracking task across 18 configurations demonstrates that LERL outperforms KCID in both tracking error and sample size reduction (Supplementary Fig. 6).



Supplementary Fig. 6: **Auxiliary policy adaptation performance across diverse configurations in bow tie tracking tasks.** Bow tie tracking performance comparisons across 18 configurations from three soft robot prototypes. Each error bar represents the full range of the tracking error, from the minimum to the maximum value.

Supplementary Figure. Fast-moving traction stick tracking performance

LERL effectively controls the tip of an elephant-trunk soft robot (configuration E_S^5-E_L^3-E_L^1) to accurately track the fast-moving traction stick (Supplementary Fig. 7), demonstrating its rapid and precise response to the stick's motion.



Supplementary Fig. 7: **Fast-moving traction stick tracking performance.** Tracking curves in the x, y, and z directions, along with the pose of the soft robot at selected moments.

Supplementary Table. Computational time per step on the Type I elephant-trunk robots

Supplementary Table 2. Computational (comp.) time per step on the Type I elephant-trunk robots. The computation is conducted on a laptop equipped with a 13th Gen Intel (R) Core (TM) i9-13900HX processor, featuring a base clock speed of 2.20 GHz.

Robot configuration	Input dimension	Lifted space dimension	Average comp. time (ms)	Max comp. time (ms)
One segment	4	24	1.862	4.004
Two segments	8	24	2.429	5.001
Three segments	12	24	3.508	8.001
Four segments	16	24	3.831	8.998
Six segments	12 (shared between segments)	24	3.917	9.001

Supplementary Table. Hyperparameters

Supplementary Table 3. Parameters for LERL and KCID

a

Parameter	Lifting dimension	Linear regression loss weight	Reconstruction loss weight	Regularization loss weight	Eigenvalue loss weight	Singular value loss weight	MLP Network Structure	Activation function	Learning rate	Optimizer	Eigenvalue threshold η_1	Singular value threshold η_2
Value	12	10	1	1	0.003	0.003	12-64-128 -64-12	ReLU	0.0003	Adam	0.5	0.2

b

Parameter	Type I robot configuration $\mathbf{E}_{\mathrm{S}}^{1}$			Type I robot configuration $\mathbf{E}_{\mathrm{S}}^{3}$–$\mathbf{E}_{\mathrm{S}}^{4}$–$\mathbf{E}_{\mathrm{L}}^{3}$			Type III robot configuration $\mathbf{E}_{\mathrm{M}}$–$\mathbf{E}_{\mathrm{L}}$	
	Target regulation	Circular tracking	Bow-tie tracking	Target regulation	Circular tracking	Bow-tie tracking	Target regulation	Circular tracking
Q	$\mathrm{diag}\{3\cdot 1_3, 1_{21}, 5\cdot 1_3, 2.5\cdot 1_3\}$	$\mathrm{diag}\{1_{24}, 4\cdot 1_3, 2\cdot 1_3\}$	$\mathrm{diag}\{1_{24}, 4\cdot 1_3, 2\cdot 1_3\}$	$\mathrm{diag}\{1_{24}, 4\cdot 1_3, 2\cdot 1_3\}$	$\mathrm{diag}\{1_{24}, 4\cdot 1_3, 2\cdot 1_3\}$	$\mathrm{diag}\{1_{24}, 4\cdot 1_3, 2\cdot 1_3\}$	$\mathrm{diag}\{1_{24}, 5\cdot 1_3, 0_3\}$	$\mathrm{diag}\{1_3, 0_3, 1_3, 0_3, 1_{12} 5\cdot 1_3, 0_3\}$
R	$\mathrm{diag}\{8\cdot 1_4\}$	$\mathrm{diag}\{5\cdot 1_4\}$	$\mathrm{diag}\{3\cdot 1_4\}$	$\mathrm{diag}\{25\cdot 1_3\}$	$\mathrm{diag}\{8\cdot 1_3\}$	$\mathrm{diag}\{8\cdot 1_3\}$	$\mathrm{diag}\{40\cdot 1_3\}$	$\mathrm{diag}\{40\cdot 1_3\}$
Discount factor	0.98							

c

Parameter	LERL with Gaussian kernel embedding	LERL with time-delay embedding
Embedding	$s = e^{-\gamma\|x-c\|^2},\ c\in[-1,1]$	$s_i = [s_i, s_{i-1}, s_{i-2}]^\top$
Q	$\mathrm{diag}\{5\cdot 1_3, 2\cdot 1_3, 1_{18}\}$	$\mathrm{diag}\{0.06\cdot 1_3, 0.03\cdot 1_9, 0.12\cdot 1_3, 0.03\cdot 1_9\}$
R	$\mathrm{diag}\{25\cdot 1_{12}\}$	$\mathrm{diag}\{5\cdot 1_{12}\}$
Discount factor	0.98	

d

Parameter	Learning rate	Batch size	ϵ_{adam}	$\beta_{1,\mathrm{adam}}$	$\beta_{2,\mathrm{adam}}$
Value	0.0005	50	1e-8	0.9	0.9999

E

Parameter	Q	R
Value	$\mathrm{diag}\{3\cdot 1_3, 1_{21}, 2\cdot 1_3, 1_3\}$	$\mathrm{diag}\{4\cdot 1_4\}$

a, Offline Koopman embedding training parameters. **b,** Online policy learning parameters for Type I and II soft robots in three configurations. **c,** Parameters for LERL with Gaussian kernel and time-delay embeddings. **d,** Adam parameters for online policy update. **e,** Parameters for KCID.
Note that $1_n = [1, \cdots, 1] \in \mathbb{R}^n$.

Supplementary Table. Comparison among three types of segments

Supplementary Table 4. Comparison among three types of segments.

Segment	Honeycomb	Muscle	Worm
Material	TPU	Rubber	PVC
Air pressure	Positive	Positive	Negative
Actuator number	4	3	3

References

1. Haggerty, D. A. *et al.* Control of soft robots with inertial dynamics. *Science Robotics* **8**, 6864 (2023).

2. Zhang, X., Pan, W., Scattolini, R., Yu, S. & Xu, X. Robust tube-based model predictive control with Koopman operators. *Automatica* **137**, 110114 (2022).

3. Lancaster, P. & Rodman, L. Algebraic Riccati equations. *Clarendon press* (1995).