

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

Corresponding Author: Professor Xinglong Zhang

This file contains all reviewer reports in order by version, followed by all author rebuttals in order by version.

Version 0:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

1. General comments:

Overall, great paper. The authors compellingly demonstrate transfer learning applied to multiple soft robot designs, which is a challenging and useful task. The included multimedia enhance the paper, and clearly show the benefits of the proposed algorithms across a wide range of robot geometries and tasks. The authors compare to prior state of the art, and the figures are of high quality. My main comment is that it is difficult to verify the authors' algorithm.

1. Limitations: How would the proposed algorithm scale to higher dimensions? I.e. more segments. Or adapt it to include reaction forces from other robots as part of a gipper or "humanoid" or octopus robot. How does the performance degrade with faster commanded accelerations? Is the trained workspace equal to each robot's actual workspace envelope?

2. Maybe you explained it somewhere and I missed... but I don't know what the exact differences are between honeycomb, muscle, and worm. To me, it looks like "Muscle" is contraction McKibben actuator, and "worm" is expanding bellows structure. And Honeycomb is also expanding type bellows actuator, but rectangular and flat pouches (rather than a circular cross-section).

3. What was special about 2k training samples? Why not 50? 250? 1000? Are there optimal types of samples / distributions that could make a better 2k set?

4. How does tracking error change with usage time? I.e. within a single test session, or across multiple test sessions.

2. Specific comments:

- The superscript and subscript meaning is only mentioned in the caption of Fig. 1. This would be nice to show in the main text along with an example like E_s^2 means small segment with wall thickness 2 mm.

3. Comments on Multimedia (Videos, etc.)

- The included multimedia clearly show the algorithm capabilities

- Overall, one thing that could help is to state the current policy goal. For example, in Supplementary Video 6, it is unclear what "Target reaching Under a payload of 1000g" means. Also in "Broken airbags", what is the robot trying to do?

(Remarks on code availability)

Many of the formulas didn't compile properly. Also it's unclear what you expect readers to be able to do. I like the transparency, but please go the next step of explaining install and usage. Also please comment the code, again for transparency and verifiability. If possible, creating a class / function diagram would also help.

Reviewer #2

(Remarks to the Author)

This paper presents a framework for controlling soft robots, utilising a policy that can adapt across multiple different configurations of soft robot with reinforcement learning utilising a linear embedding space. The mapping of shared dynamics between the different configurations in this linear embedding space along with online policy adaptation allows the framework to quickly transfer policies from one configuration to another without the need to train each one from scratch.

There are a total of 30 configurations of elephant trunk, soft-muscle, and worm robots with varying wall thicknesses which affect their stiffnesses and three different lengths. The results demonstrate the performance of the framework on a diverse set of tasks which include hammering nails, serving drinks, calligraphy writing, and a hand-eye reaction game. It's a well written paper, but nevertheless it can be improved as follows:

1. Unless I have overlooked it, there seems to be no mention of any of the materials used for the soft robots. If they are in fact stated elsewhere then consider putting them in the Methods section where each of the types is detailed.
2. The images displaying the different types of platforms are very informative, but there do not seem to be any dimensions listed. It would be helpful to specify the centerline lengths of the segments, as well as the range of wall thicknesses used for the honeycomb segments.
3. The criteria for selecting the 30 configurations that are used do not seem to be discussed. Please provide some context or rationale behind these choices that lead to these specific configurations.
4. Koopman control method with inertial dynamics (KCID) is defined in the Results section but is referenced prior to it in the caption of Table 1.
5. KCID seems to be referred to as KLQR (which comes from the author of Ref. 7) in Fig. 9 and Fig. 10 but not their captions. Please maintain consistency.
6. Consider adjusting line thickness in figures such as Fig. 3c and Fig. 6e to improve readability.
7. The naming convention for the muscle and worm types are identical, though the accompanying images of the actual assemblies do clarify which type is used. For more clarity, consider modifying the notation, perhaps including an additional subscript denoting which type is used.

(Remarks on code availability)

Reviewer #3

(Remarks to the Author)

This work presents an approach for soft robotics control through the use of reinforcement learning in a Koopman embedding space. They claim a 75x reduction in training samples needed compared to state-of-the-art methods without significant impact on control optimization. Further, it is claimed that this model is applicable to diverse soft robotic morphologies including variations in materials, stiffnesses, structural geometry, and assemblies.

The significance of this work does not rely on the novel use of Koopman operators or reinforcement learning in soft robotics. Its novelty is in the marriage between the two and the use of a Koopman embedding space to streamline and generalize learned control methods. It is this reviewer's belief that this combination is a currently under-explored method for generating control policies for soft robotics. However, other work in similar conceptual spaces (reinforcement learning married to Koopman embedded space for the control of soft/continuum robots) does exist [1-3]. The novelty of this work seems to be primarily in the validation tasks (hammering, serving drinks, calligraphy, etc) using this control paradigm. As such, additional data and analysis of these tasks may be helpful to better support this work. One of the main claims of the work is that there is a 75x reduction in training samples required compared to other methods. How does the learned control policy compare to state-of-the-art control policies on these tasks when given varied amounts of training data? What is the failure rate, if any, of the robot when performing these tasks? It seems that the claim of 75x less training data required is a major part of this work, however comparisons to other methods and validations of this claim seem to be misrepresentative. From the work itself: "ERL using 22k training samples, 20k for offline Koopman embedding training and 2k for policy generation." Indicating that 20k samples are used for training the Koopman embedding space and an additional 2k are used to train the control policy. In table 1, the learning methods used in this work are compared directly to other learning methods, however, the compared learning methods have all of their training data samples bundled together as opposed to just the policy generation samples (at least as far as I can tell from reading the cited works). This drastically under-represents the samples required for this work and when taken into account puts it in line with many state-of-the-art methods for control policy generation. I suggest reviewing table 1 and putting in a disclaimer as to why only the policy generation training data is accounted for in this work as opposed to all training data required for this combined learning and Koopman operator approach. Further, I would go over the table again and ensure it is a fair comparison, especially with the other methods using Koopman operators (make sure that you are not counting their embedded space training data if you are not including your own). With this view on training data, the performance of this model is still greater than many state-of-the-art methods, however it requires substantially more training data than is eluded to in the claims and tables of this work.

Another claim is that the control policies work across 30 soft robot configurations, however the notability of these configurations is somewhat suspect. There are three main soft robotic assemblies used to validate the control policy, all three are continuum robots utilizing pneumatic actuation systems. The primary variation in the robots themselves is their stiffness and the amount of connected segments or blocks they use (which can be constructed of varying assemblies). Overall, these configurations all have relatively similar movements and the control methods used are very similar. I think this claim may need to be reviewed as it seems to over-inflate the applicability of the learned policy when it is being applied to functionally very similar mechanisms.

The methodology is sound and the results seem to be genuine. There also appears to be enough data and detail provided for the methods of this work to be reproduced.

Additionally, many figures in this work are unclear and cluttered with text size dropping substantially below the default font size used in this work. General cleanup and reformatting of the figures for clarity may significantly enhance this work.

This reviewer recommends revision and additional clarity into the comparison methods used to justify claims made in this work.

References:

[1] Sinha, Anirban, and Yue Wang. "Koopman operator-based knowledge-guided reinforcement learning for safe human-robot interaction." *Frontiers in Robotics and AI* 9 (2022): 779194.

[2] Ji, Guanglin, et al. "Efficient real2sim2real of continuum robots using deep reinforcement learning with Koopman operator." *IEEE Transactions on Industrial Electronics* (2025).

[3] Abraham, Ian, and Todd D. Murphey. "Active learning of dynamics for data-driven control using Koopman operators." *IEEE Transactions on Robotics* 35.5 (2019): 1071-1083.

(Remarks on code availability)

Version 1:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

The authors have a clearly organized response to all of the reviewers, which is appreciated. I only have a few minor comments lingering.

- Fig. 4d second y axis is inside of subfigure 4e.

- 4a not clear what is gray vs. teal, unless we are supposed to assume the color code in d, which doesn't make much sense. Based on the description, I understand it to be gray is varying the feedforward training samples, and teal is varying the transfer sample size.

(Remarks on code availability)

I did review the Git Repository and it seems well documented.

Reviewer #2

(Remarks to the Author)

The authors have addressed my comments properly. I don't have any other comments and I can recommend the publication of the paper.

(Remarks on code availability)

Reviewer #3

(Remarks to the Author)

The authors have carefully addressed all of my previous comments. The additional analyses and clarifications have strengthened the manuscript, and I am satisfied with the revisions. The additional robot configurations and enhanced clarity of figures add substantially to this work. Additionally, the improved clarity on how LERL is compared to other methods is appreciated. It appears that all comparisons are direct and do not misrepresent critical metrics. I have no further comments and recommend acceptance in its current form.

(Remarks on code availability)

Open Access This Peer Review File is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

In cases where reviewers are anonymous, credit should be given to 'Anonymous Referee' and the source.

The images or other third party material in this Peer Review File are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

To view a copy of this license, visit <https://creativecommons.org/licenses/by/4.0/>

Response to the Review Comments

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

**Manuscript Reference Number:
ID NCOMMS-25-60780-T**

Date: January 22, 2026

Response To Reviewer #1

Overall Comment

1. Overall, great paper. The authors compellingly demonstrate transfer learning applied to multiple soft robot designs, which is a challenging and useful task. The included multimedia enhance the paper, and clearly shows the benefits of the proposed algorithms across a wide range of robot geometries and tasks. The authors compare to the prior state of the art, and the figures are of high quality. My main comment is that it is difficult to verify the authors' algorithm.

Response

Thank you very much for the positive and encouraging assessment of our work. In the newly uploaded manuscript, we have made the following revisions to improve the clarity of the implementation. The specific modifications are as follows:

- We have updated the detailed manual for the uploaded source code (see also the GitHub repository <https://github.com/xinglongzhangnurd/LERL-for-soft-robots>), which includes a function diagram to show the code structure, implementation procedures covering Koopman embedding training, policy training and transfer. In addition, we have added comments to the core functions in the source code.
- We have added more implementation details for the pseudocode of LERL in Supplementary Section 4, including the network structures of the Koopman embedding and the feedforward policy, as well as the specific input excitations in training data collection.

Reviewer Comment

2. Limitations: How would the proposed algorithm scale to higher dimensions? I.e. more segments. Or adapt it to include reaction forces from other robots as part of a girpper or "humanoid" or octopus robot. How does the performance degrade with faster commanded accelerations? Is the trained workspace equal to each robot's actual workspace envelope?

Response

Thank you for your insightful comment. Below, we address your concerns one by one.

-How would the proposed algorithm scale to higher dimensions? I.e. more segments.

The scalability of our approach to more dimensions is due to the following two aspects. Firstly, the control policy is a feedback policy based on the robot tip state, defined as $x = [p_x, p_y, p_z, v_x, v_y, v_z, \theta_x, \theta_y, \theta_z, w_x, w_y, w_z]^T \in \mathbb{R}^{12}$. The dimension of x is independent of the number of segments in the soft robot. Moreover, the Koopman embedding is implemented as a multilayer perceptron (MLP) with x as input for all configurations. This design enables the Koopman embedding trained on one segment to be shared across robot configurations with diverse segments, and even extended to heterogeneous and hybrid-assembled robots, thereby facilitating rapid online model-free policy learning within the shared linear embedding space. Secondly, although the number of control channels increases with the number of segments, we employ a padding procedure (Supplementary Section 5 in the revised manuscript) to initialize the control policy when

transferring to robot configurations with diverse segments, i.e., to a higher-dimensional control space. In particular, this approach replicates the partial policy gain matrix learned in the lower-dimensional space to populate the newly added control channels, which effectively bridges the dimensionality gap in practice and enables successful policy transfer across robots with varying numbers of segments and control dimensions (see Figs. 4, 11-12 in the revised manuscript for the transfer performance across 33 robot configurations).

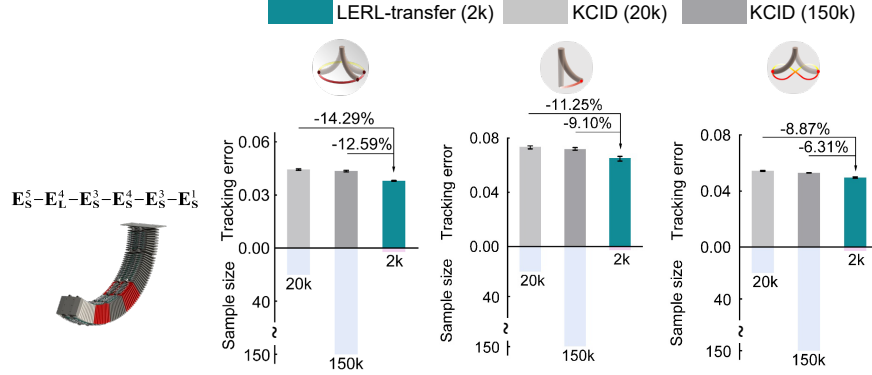


Fig. 1: Performance validation of LERL on a Type I elephant-trunk robot configuration with six segments. LERL reduces tracking errors by 12.59%, 9.1%, and 6.31% in target reaching, circular, and bow tie tracking tasks compared to KCID with 150k samples, similar to the results obtained for configurations with two, three, and four segments.

To further validate the effectiveness of our approach to robots with more segments (higher dimensions), we have conducted additional experimental validations using a Type I elephant-trunk robot configuration with six segments $E_S^5-E_L^4-E_S^3-E_S^4-E_S^3-E_S^1$ (greater than four in the previous version). Note that, in principle, a six-segment Type I robot requires 24 proportional valves (i.e., 24 control channels), since each elephant-trunk segment is actuated by four valves. However, the Type I platform (Fig. 7 in the revised manuscript) provides only 16 D/A channels. To accommodate this hardware constraint, the control inputs are shared in the experiment between adjacent segments: segments 1–2, 3–4, and 5–6, each pair shares the same control inputs. This makes dynamic control of the six-segment configuration more challenging, as fewer actuators are used due to the shared control channels. The experimental results have been included in Fig. 4, Figs. 11-12 of the revised manuscript (see Fig. 1 above for a combination of the above results). The results demonstrate that LERL reduces tracking errors by 12.59%, 9.1%, and 6.31% in target reaching, circular, and bow tie tracking tasks when compared to the state-of-the-art KCID (with 150k samples) on this six-segment configuration, similar to the results obtained for configurations with two, three, and four segments (see Fig. 4a, Figs. 11-12 of the revised manuscript). These results validate that LERL could scale to higher-dimensional configurations.

- Or adapt it to include reaction forces from other robots as part of a gripper or “humanoid” or octopus robot.

Thanks for this brilliant idea. Our soft robot can be integrated as a part or an end-effector within heterogeneous robotic systems, such as grippers, humanoid robots, and octopus-inspired robots, the latter of which often exhibit parallel structural architectures. Inspired by your comment, we designed a new parallel-series hybrid robot configuration, $[E_M^1|E_M^1|E_M^1]-O_M$, which introduces a parallel module composed of three elephant-trunk segments connected in parallel (see Fig. 2 below). This parallel module is further connected in series with a worm-like segment, forming a hybrid parallel-series structure. This configuration is designed to simulate the interactions between a soft arm and other heterogeneous robotic components,

such as those found in octopus-inspired systems, which typically feature parallel structures.

We have conducted additional experiments on this parallel-series hybrid robot. As shown in Fig. 2 below (corresponding to Fig. 4d and subfigures of Figs. 11-12 of the revised manuscript), LERL can effectively adapt to this heterogeneous configuration with unknown internal reaction force couplings through online, model-free policy learning. Compared to KCID trained with 150k samples, LERL reduces tracking errors by 7.43%, 7.18%, and 22.24% in circular tracking, target reaching, and bow tie tracking tasks, respectively, achieving performance consistent with that observed on previously studied configurations.

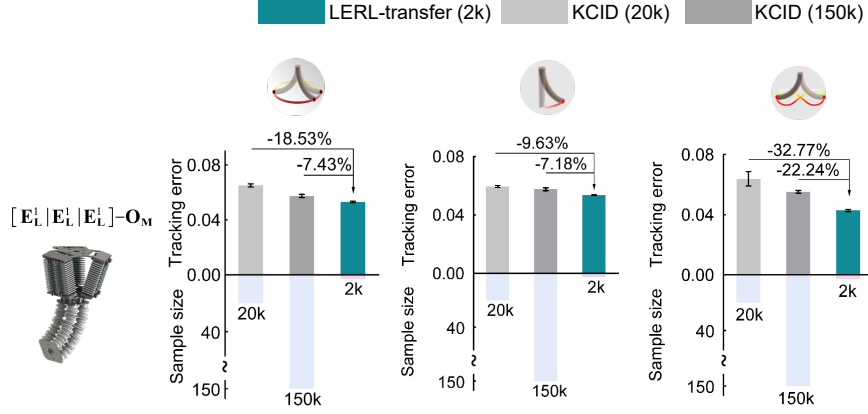


Fig. 2: Performance validation of LERL on a new parallel-series hybrid robot configuration. LERL reduces tracking errors by 7.43%, 7.18%, and 22.24% in circular tracking, bow tie tracking, and target reaching tasks compared to KCID with 150k samples, achieving performance consistent with that observed on previously studied configurations.

Furthermore, to simulate external forces, we have used a rigid bar to apply instantaneous intervention to the parallel-series hybrid robot at four different positions, as shown in Fig. 3c below (corresponding to Fig. 10c in the revised manuscript). The results demonstrate that the LERL policy transferred to this parallel-serial robot remains robust to external interventions, enabling the system to quickly recover and achieve stable tracking after each intervention.

The above statements are also added on Page 18 of the revised manuscript and listed below.

“ Fig. 10c also presents tracking error curves on a heterogeneous parallel-series hybrid robot, $[E_L^1|E_L^1|E_L^1]-O_M$, under external interventions applied at four different locations. The results demonstrate that the LERL policy transferred to this parallel-serial robot remains resilient to external force disturbances, enabling the system to quickly recover and achieve stable tracking after each intervention (Supplementary Video 6). ”

- How does the performance degrade with faster commanded accelerations?

We have also conducted experiments to investigate the performance of LERL under different commanded accelerations, ranging from $3.62 \text{ mm} \cdot \text{s}^{-2}$ to $82.14 \text{ mm} \cdot \text{s}^{-2}$ across six levels (Fig. 3b below and Fig. 10b in the revised manuscript; Supplementary Video 6). The results demonstrate that LERL could stably handle tasks under varying commanded accelerations, although its performance degrades slightly at higher acceleration levels. In contrast, the model-based KCID fails to complete the tracking task when the commanded accelerations exceed $16.51 \text{ m} \cdot \text{s}^{-2}$. This superiority of LERL over KCID is due to its online fast policy learning capability to account for dynamic variations.

The above statements are included on Page 18 of the revised manuscript and also listed below.

“ We then compared LERL with KCID in a circular tracking task under varied accelerations, ranging from $3.62 \text{ mm} \cdot \text{s}^{-2}$ to $82.14 \text{ mm} \cdot \text{s}^{-2}$ across six levels (Fig. 10b, Supplementary Video 6). The results demon-

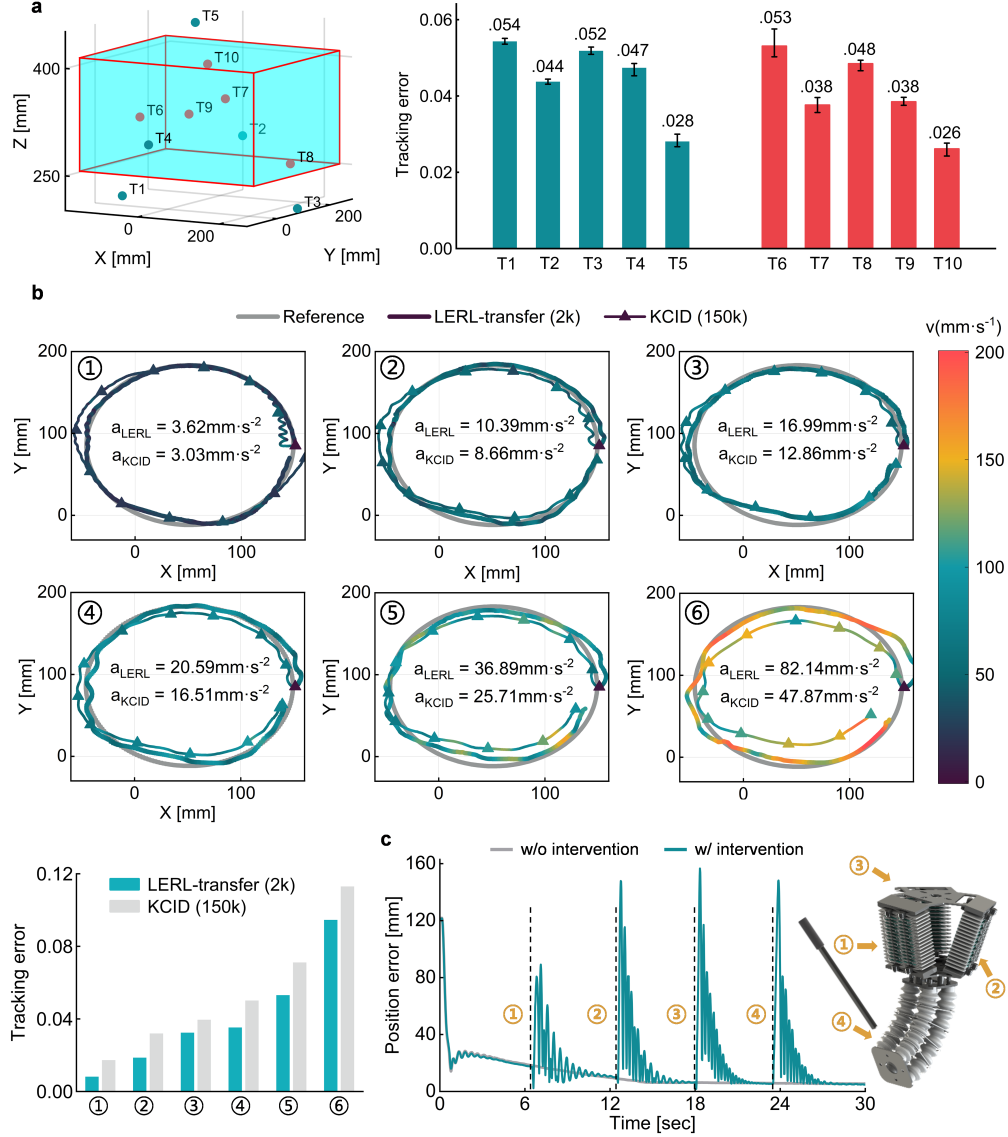


Fig. 3: Auxiliary performance and robustness evaluations. **a**, Performance comparison on target positions within and outside of the training space. **b**, Performance in the circular tracking task under varied accelerations. **c**, Tracking error curves under external interventions applied at four different positions.

strate that LERL experiences slight performance degradations when acceleration grows and outperforms KCID at all acceleration conditions.”

- Is the trained workspace equal to each robot’s actual workspace envelope?

Our method aims to use a limited amount of data to train the Koopman embedding and control policy. Consequently, the trained workspace for each configuration might be smaller than the associated actual workspace envelope. In regions outside the trained workspace, the robot may encounter previously unseen dynamic variations, which could lead to performance degradation with direct policy deployment. However, this limitation is mitigated by our approach, which can rapidly adapt to such changes through online

learning, maintaining stable performance even beyond the trained workspace. Furthermore, as theoretically shown in Supplementary Section 8, our method guarantees robust control performance and outperforms the corresponding model-based design under mild Lipschitz continuous conditions.

Specifically, in the revised manuscript, we have evaluated the performance of LERL on target reaching for the robot configuration $E_S^4-E_L^4-E_L^3$, with target positions outside the training workspace. Ten target positions were sampled along five representative directions, including five targets (T1–T5) outside the trained workspace and five targets (T6–T10) within the trained workspace. The results reported in Fig. 3a above (corresponding to Fig. 10a in the revised manuscript) show that LERL can reach targets outside the trained workspace with only slight performance degradation, demonstrating its generalizability to unseen regions through fast adaptation via online policy updates.

The above statements are included on Page 18 in the revised manuscript and also listed below.

“We evaluated the performance of LERL on target positions outside of the training space on a robot configuration $E_S^4-E_L^4-E_L^3$. Specifically, we sampled 10 target positions in five representative directions, where each direction includes a pair of targets: one outside the trained workspace and one inside it. As shown in Fig. 10a, T1–T5 are beyond the trained workspace, and T6–T10 are their corresponding counterparts within the trained workspace. The experimental results show that the tracking errors for targets T1–T5 are slightly larger than those for the corresponding targets within the trained workspace (T6–T10). These findings confirm the generalizability of LERL to unseen regions due to the fast adaptability through online policy updates.”

Reviewer Comment

3. Maybe you explaiend it somewhere and I missed... but I don't know what the exact differences are between honeycomb, muscle, and worm. To me, it looks like "Muscle" is contraction McKibben actuator, and "worm" is expanding bellows structure. And Honeycomb is also expanding type bellows actuator, but rectangular and flat pouches (rather than a circular cross-section).

Response

The three types of robots differ in materials, structures, control dimensions, and actuation types (positive or negative).

- The elephant-trunk-shaped arm is designed with a modular honeycomb segment architecture and 3D-printed from thermoplastic polyurethane (TPU). Each segment has four control channels actuated by positive air pressure via inflatable airbags, enabling both deflection and extension of the arm.
- Each muscle segment consists of three parallel pneumatic muscles made from rubber, with three control channels actuated by positive air pressure to drive segment deflection.
- The worm-like segment comprises three parallel bellows 3D-printed from polyvinyl chloride (PVC). Each segment has three control channels actuated by negative air pressure, producing dynamic arm deflection.

In the revised manuscript, we have added the following content in the Methods Section (Page 12) and Table 1 below (Supplementary Table 3 on Page S15 of the revised manuscript) to clarify the above differences.

“The Type I soft robotic platform features an elephant-trunk-shaped arm with a modular honeycomb segment architecture, 3D-printed from the thermoplastic polyurethane (TPU) material.”

Table 1: Comparison among three types of segments.

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

“Each segment of the Type II robot consists of three parallel pneumatic muscles created by the rubber material. ”

“Type III robots use a worm-like segment structure 3D-printed from the polyvinyl chloride (PVC) material,...”

Reviewer Comment

4. What was special about 2k training samples? Why not 50? 250? 1000? Are there optimal types of samples/distributions that could make a better 2k set?

Response

Thank you for raising this important question.

The 2k dataset contains 500 samples for offline feedforward policy training, and 1.5k samples used for online policy learning during control implementation. According to your comment, we first conducted experiments using 50, 250, and 500 offline samples to train and evaluate the feedforward policy. We then evaluated transfer performance with 1k, 2k, 3k, and 5k total samples, where 500 samples were allocated to feedforward policy training, and the remainder were used for online policy learning. The results in Fig. 4a below (corresponding to Fig. 9a in the revised manuscript) show that the feedforward policy with 500 offline samples substantially outperforms those with 50 and 250 samples. With 500 offline samples, adding 1.5k online interaction data could lead to substantial performance improvements. In contrast, further increasing the dataset size beyond 2k samples yields only marginal gains. This suggests that 2k samples (500 offline and 1.5k online) constitute an appropriate dataset size for policy transfer across robot configurations.

Note also that, the 500 offline feedforward samples could be generated with different input excitations. In the original version, we collected the 500 offline feedforward samples using only sinusoidal input excitation. In the revised manuscript, we additionally collected offline feedforward data using random input excitation to enable a direct comparison. The comparative results shown in Fig. 4b below (Fig. 9b in the revised manuscript) demonstrate that the feedforward policy trained with sinusoidal excitation outperforms that trained with random input excitation. Notably, this performance advantage persists even when online learning is incorporated. The reason for this result is that random input excitation could induce a twisted motion status of the soft robotic arm, leading to multiple steady-state control inputs corresponding to the same robot tip state. This non-uniqueness introduces ambiguity during feedforward policy training. In contrast, sinusoidal input excitation mitigates this effect by promoting smoother and more consistent arm motions.

The above statements have been included on Page 18 in the revised manuscript and listed below.

“ We investigated the dataset composition required to enable effective policy transfer and learning across different robot configurations. Specifically, we first conducted experiments using 50, 250, and 500 offline samples to train and evaluate the feedforward policy. We then evaluated transfer performance with 1k, 2k, 3k, and 5k total samples, where 500 samples were allocated to feedforward policy training, and the

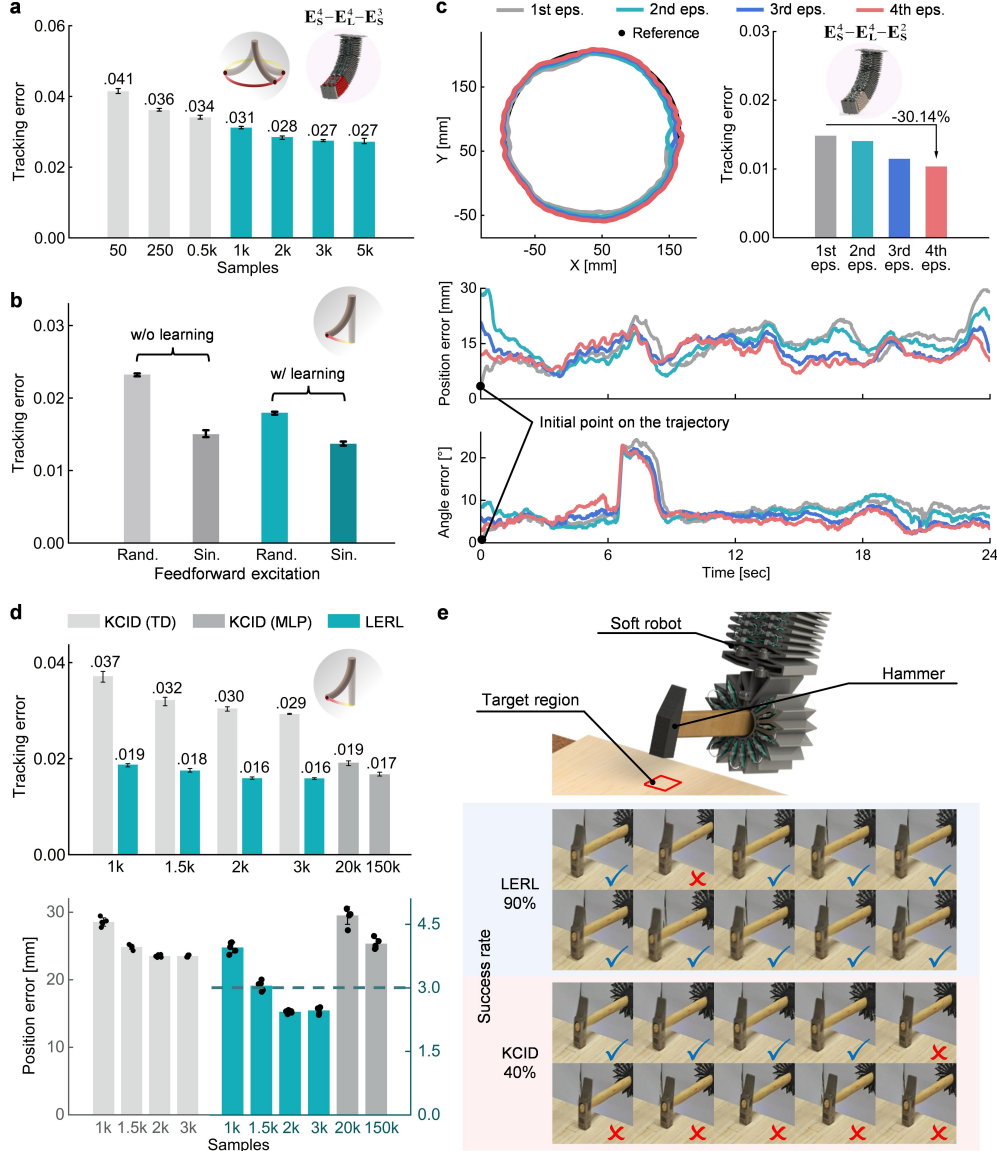


Fig. 4: Auxiliary performance evaluations of dataset properties and success rate. **a**, Transfer performance under different dataset sizes and compositions. **b**, Task performance under feedforward policies trained with datasets generated using sinusoidal and random input excitations. **c**, Four-episode (eps.) learning performance in circular tracking of one elephant-trunk robot. **d**, Transfer performance comparison under varied amounts of training data. **e**, Success rate comparison for hammering a fixed point within a predefined positional tolerance of 3 mm.

remainder were used for online policy learning. The results in Fig. 9a show that the feedforward policy offline trained with 500 samples substantially outperforms those with 50 and 250 samples. With 500 offline samples, adding 1.5k online training interaction data could lead to substantial performance improvements. In contrast, further increasing the dataset size beyond 2k samples yields only marginal gains. This suggests that 2k samples (500 offline and 1.5k online) constitute an appropriate dataset size for policy transfer across robot configurations.”

“We also investigated how different sampling distributions influence control performance. Since the 1.5k

online interaction data of 2k may not follow a predefined distribution, we utilize two excitation schemes, i.e., sinusoidal and random control inputs, to collect the 0.5k quasi-static data samples for feedforward policy training (Supplementary Section 2). The comparative results in Fig. 9b show that the feedforward policy trained with sinusoidal input excitation outperforms that trained with random input excitation, and this advantage persists even in online learning scenarios. The reason is that random input excitation could induce a twisted motion status of the soft robotic arm, leading to multiple steady-state control inputs corresponding to the same robot tip state. This non-uniqueness introduces ambiguity during feedforward policy training. In contrast, sinusoidal input excitation mitigates this effect by promoting smoother and more consistent arm motions.”

Reviewer Comment

5. How does tracking error change with usage time? I.e., within a single test session, or across multiple test sessions.

Response

We appreciate your inspiring questions. We have conducted additional experiments to demonstrate that the tracking error decreases over time. As shown in the additional experiments in Fig. 4c above (corresponding to Fig. 9c in the revised manuscript), both the position and orientation (Euler-angle) errors exhibit an overall decreasing trend across the four test sessions, and the tracking error in the 4th episode is reduced by 30.14% compared with that in the 1st episode.

Note that a noticeable fluctuation (approximately 10°) in the orientation (Euler-angle) error is observed during the 6–9 s time interval, whereas no such fluctuation appears in the position error. This is likely caused by the motion-capture system temporarily losing one marker point during tracking, which can slightly perturb the estimated end-effector orientation while leaving the Cartesian position estimate unaffected.

The associated results and analysis are included on Page 18 and listed below.

“The online multi-episode learning ability of our method was validated with elephant-trunk and worm robots. Our approach enables the elephant-trunk robot in configuration $E_S^4-E_L^4-E_S^2$ to accurately track bow tie (Fig. 6e) and circular (Fig. 9c) trajectories. In addition, the tracking errors decrease with each learning episode and are reduced by 22.61% and 30.14% after four learning episodes for the bow tie and circular trajectories, respectively, verifying the online learning capability.”

Reviewer Comment

6. Specific comments. The superscript and subscript meaning is only mentioned in the caption of Fig. 1. This would be nice to show in the main text, along with an example like E_S^2 means small segment with wall thickness 2 mm.

Response

Thank you for your helpful suggestion to clarify the meaning of superscripts and subscripts. Following your suggestion, we have added the meaning of the superscript and subscript along with the appearance of E_S^1 , where the superscript “1” indicates the lowest stiffness level determined by a honeycomb hole of 45 mm, as shown in Fig. 5 below and Table 2 below (Corresponding to Supplementary Fig. 3 and Supplementary Table 1 in the revised manuscript).

We have added the following content to the revised manuscript.

On Pages 6-7: “The initial step involves pretraining a multilayer perceptron (MLP)-based Koopman embedding $\Psi(x)$ using input-state data collected from a single honeycomb segment (E_S^1 , where ‘S’ represents a short segment, and ‘1’ denotes stiffness level 1 as shown in Fig. 1, Supplementary Fig. 3, and Supplementary Table 2, wherein $x \in \mathbb{R}^{12}$ represents the pose (position and orientation) and velocity (linear and angular velocities) of the robot tip (Methods and Supplementary Section 1).”

In Supplementary Section 9: “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 Supplemental Table 1.”

Table 2: 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

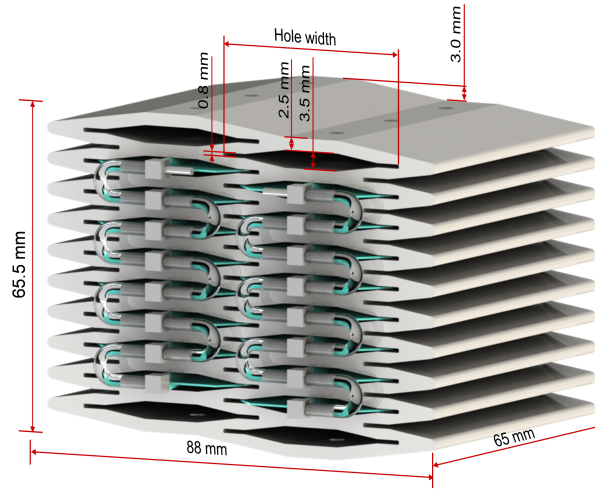


Fig. 5: Schematic diagram of the honeycomb segment structure. The stiffness level of the honeycomb segment can be adjusted by modifying the hole width.

Reviewer Comment

7. *Comments on Multimedia (Videos, etc.) The included multimedia clearly shows the algorithm capabilities - Overall, one thing that could help is to state the current policy goal. For example, in Supplementary Video 6, it is unclear what "Target reaching Under a payload of 1000g" means. Also in "Broken airbags", what is the robot trying to do?*

Response

Thank you for your comment, which is valuable in improving the quality of the videos. In the revised manuscript, we have added more captions to clearly state the goals in the videos, listed below.

- Supplementary Video 6: We added the caption “Performance in target reaching task under varied payloads” to clarify that the experiment evaluates target reaching performance under different payload conditions. We also added the captions “Robustness validation under broken airbags” and “Leaking airbags blow the paper up” to illustrate robustness evaluation of LERL under damaged airbags. In addition, the caption “Resilience of the parallel-series hybrid robot under external intervention” was added for the newly introduced experiments.
- Supplementary Video 1: We added the caption “The robot then used the hammer to quickly drive nails into both sides of the wooden board, completing the task in 72 seconds” to clarify the task objective, and included experimental results comparing the success rates of LERL and KCID.
- Supplementary Video 2: We added the caption “LERL policy enabled the elephant-trunk robot to perform the drinks-serving task with precise and smooth operation, emulating the actions of a skilled human bartender” at the beginning of the video to clearly present the task objective.
- Supplementary Video 3: We added the caption “LERL policy enables the hybrid-assembled soft robot to achieve smooth handwriting” to indicate the goal of the task.
- Supplementary Video 4: We added the caption “The soft robot uses a camera to detect sticks falling randomly from a height of 0.48 meters and quickly moves its net pocket to catch a falling stick within 100 ms” to clarify the task objective.
- Supplementary Video 5: We updated the captions to “Policy adaptation to soft-muscle and hybrid-assembled robots” and “Policy adaptation to worm robots and hybrid-assembled robots” to reflect the newly added experiments.

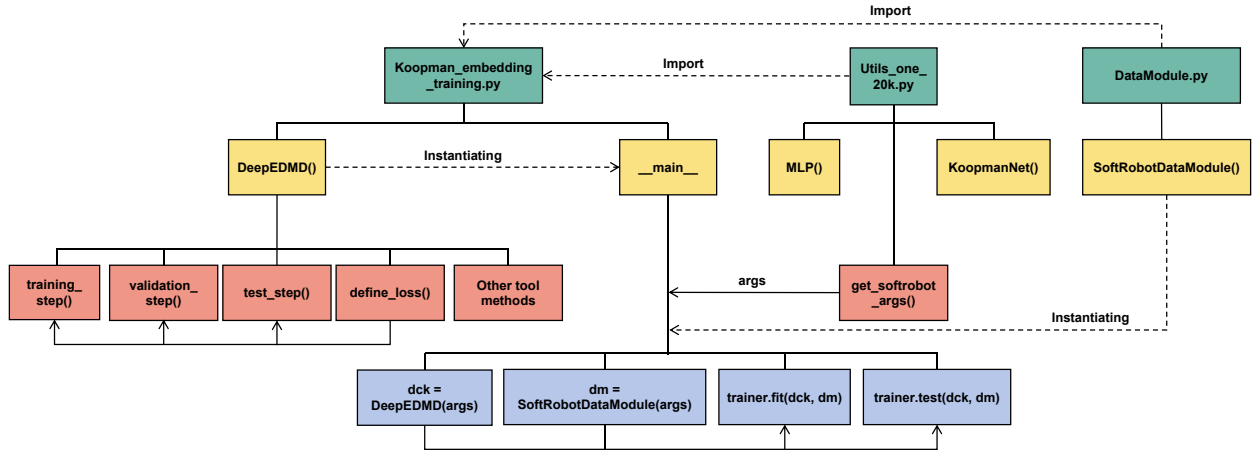


Fig. 6: Function diagram of “Koopman_embedding_training.py”.

Reviewer Comment

8. (Remarks on code availability) Many of the formulas didn't compile properly. Also it's unclear what you expect readers to be able to do. I like the transparency, but please go the next step of explaining install and usage. Also please comment the code, again for transparency and verifiability. If possible, creating a class/function diagram would also help.

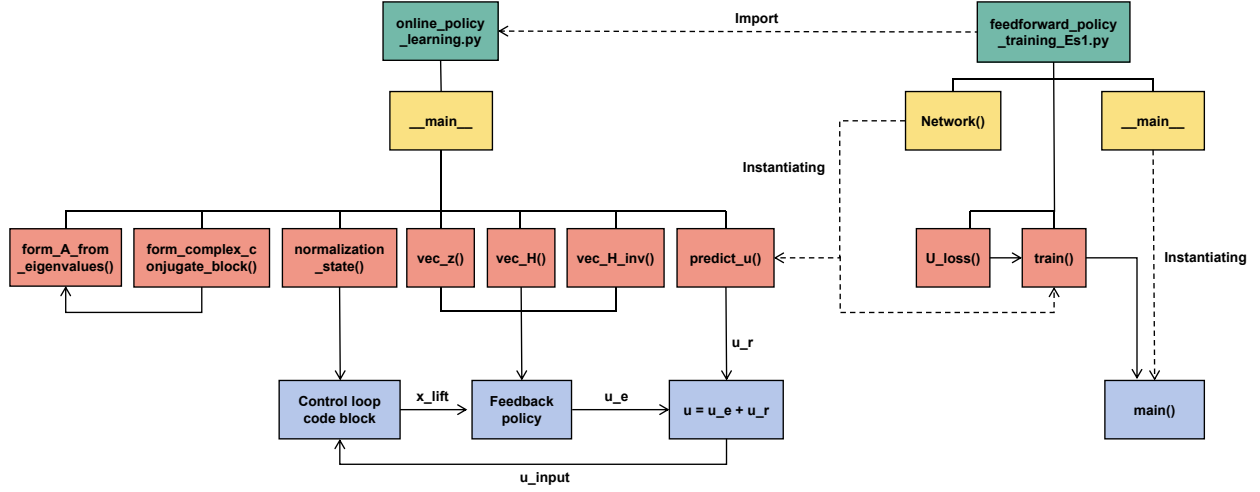


Fig. 7: Function diagram of “online_policy_learning.py” and “feedforward_policy_training_Es1.py”.

Response

Thanks for the comment. In the revised manuscript, we have made the following modifications to make the implementation of the code clearer.

- We have updated the detailed manual for the uploaded source code (see also the GitHub repository https://github.com/xinglong_zhangnurd/LERL-for-soft-robots), which includes function diagrams to show the code structure (see Figs. 6-7 above), implementation procedures covering Koopman embedding training, policy training and transfer. In addition, we have added comments to the core functions in the source code.
- We have added more implementation details for the pseudocode of LERL in Supplementary Section 4, including the network structures of the Koopman embedding and the feedforward policy, as well as the specific input excitations of training data collection.

In the uploaded codes, the file “Koopman_embedding_training.py” is ready for training the Koopman embedding. The file “feedforward_policy_training_Es1.py” is ready for training the feedforward policy of E_S^1 . The file ‘online_policy_learning.py’ implements the online policy learning module, which updates the policy through interaction with the physical hardware. Note that the policy learning algorithm is directly implemented on the real-world robotic platform. In addition to the reported details of our robotic platform, we have also released the 3D models of the three robot segments, one for each segment type, to enhance the reproducibility of our work. Using these models, one can fabricate and assemble the soft robots with a standard 3D printer or adjust the design parameters to create customized robot structures.

Response To Reviewer #2

Overall Comments

This paper presents a framework for controlling soft robots, utilising a policy that can adapt across multiple different configurations of soft robot with reinforcement learning utilising a linear embedding space. The mapping of shared dynamics between the different configurations in this linear embedding space along with online policy adaptation allows the framework to quickly transfer policies from one configuration to another without the need to train each one from scratch. There are a total of 30 configurations of elephant trunk, soft-muscle, and worm robots with varying wall thicknesses which affect their stiffnesses and three different lengths. The results demonstrate the performance of the framework on a diverse set of tasks, which include hammering nails, serving drinks, calligraphy writing, and a hand-eye reaction game. It's a well written paper, but nevertheless it can be improved as follows.

Response

Thank you so much for your positive evaluation. In the revised manuscript, we have carefully addressed all your concerns to improve our work. The main revisions are summarized below.

- We presented the criteria for selecting the 30 robot configurations (extended to 33 configurations in the revised manuscript).
 - We clarified the specific materials used to construct the three types of soft robots, and provided additional details on each segment, including control dimensions, centerline length, and stiffness level.
 - We introduced a new naming convention for different types of soft robots to enhance clarity.
 - We corrected issues of abbreviations, inconsistencies, and figures' line thickness.
-

Reviewer Comment

1. Unless I have overlooked it, there seems to be no mention of any of the materials used for the soft robots. If they are in fact stated elsewhere then consider putting them in the Methods section where each of the types is detailed.

Response

Thanks for your comment. We have added the materials used for each type of robot in the Methods section (Page 12) of the revised manuscript. The corresponding statements are cited below for convenience and are also reflected in Table 3 below (corresponding to Supplementary Table 3 on Page S15 of the revised manuscript).

“The Type I soft robotic platform features an elephant-trunk-shaped arm with a modular honeycomb segment architecture, 3D-printed from the thermoplastic polyurethane (TPU) material.”

“Each segment of the Type II robot consists of three parallel pneumatic muscles created by the rubber material.”

“Type III robots use a worm-like segment structure 3D-printed from the polyvinyl chloride (PVC) material,.... ”

Table 3: Comparison among three types of segments.

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

Reviewer Comment

2. The images displaying the different types of platforms are very informative, but there do not seem to be any dimensions listed. It would be helpful to specify the centerline lengths of the segments, as well as the range of wall thicknesses used for the honeycomb segments.

Response

Thank you for your detailed comment. In the revised manuscript, we have added the input dimensions and centerline lengths of the short, medium, and long segments for the three types of robots to the caption of Fig. 8 below (Fig. 1 of the revised manuscript). The associated statements are also listed below.

“Type I elephant-trunk robots: Each segment features a honeycomb structure with four pneumatic actuators. Honeycomb segments’ properties vary with the hole width between walls (superscript indices $1, \dots, N$ denote the stiffness levels determined by the honeycomb holes detailed in Supplementary Fig. 3 and Supplementary Table 1) and segment lengths (subscript indices ‘S, M, L’ represent segment lengths: Short, Medium, and Long). The centerline lengths for short and long honeycomb segments are 65.5 mm and 101.5 mm, respectively.”

“Type II soft-muscle robots: Each segment is composed of three parallel pneumatic muscles, each driven by a pneumatic actuator. The centerline lengths of the short, medium, and long muscle segments are 159 mm, 194 mm, and 229 mm, respectively.”

“Type III worm robots: Each segment features a worm structure with three pneumatic actuators. The short, medium, and long worm segments have centerline lengths of 190 mm, 220 mm, and 270 mm, respectively.”

Table 4: 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

We also acknowledge that using the wall thickness to denote the stiffness level of a segment may not be sufficiently rigorous. In the revised manuscript, we have clarified in Fig. 1 of the revised manuscript and at the first occurrence of the honeycomb segment E_S^1 that 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. Detailed definitions of the honeycomb hole geometry and the design parameters for each stiffness level are provided in Table 4 above and Fig. 9 below (Supplementary Fig. 3 and Supplementary Table 1 in the revised manuscript).

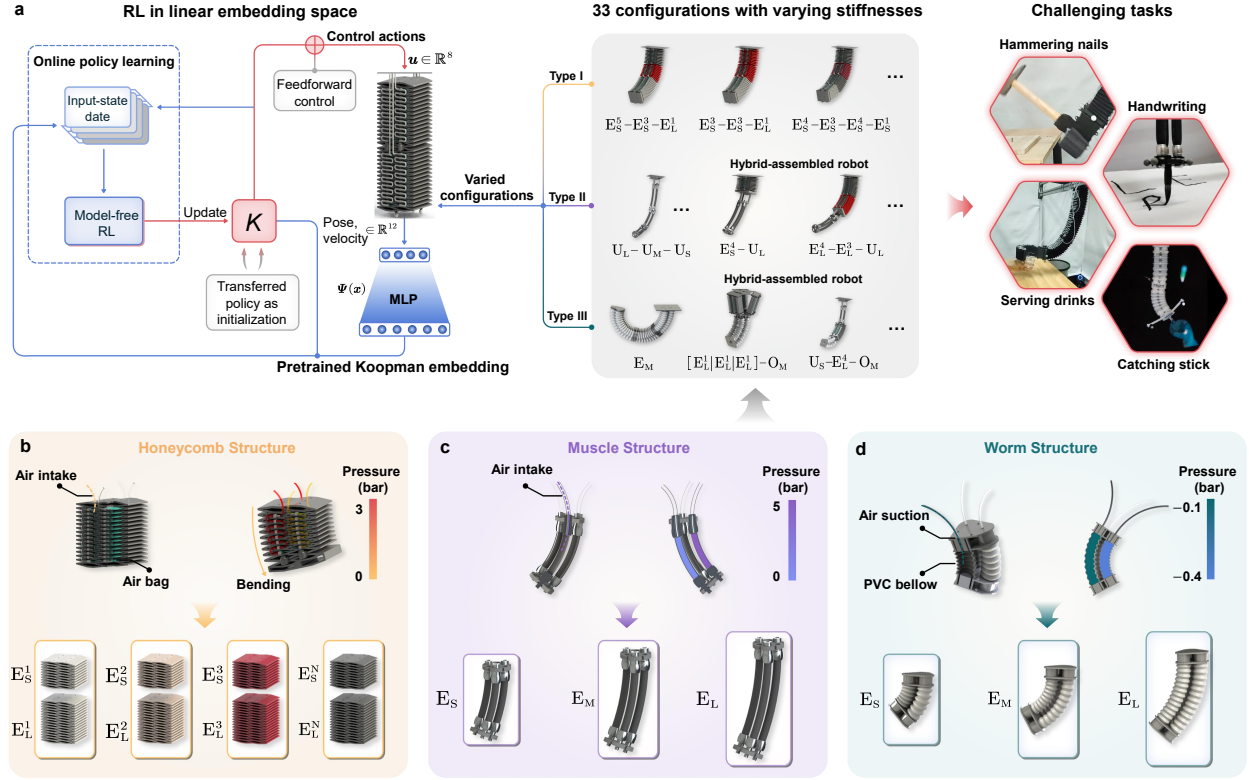


Fig. 8: Our generalizable control system across 33 configurations with diverse materials and stiffnesses.

a, A generalizable control system with fast adaptability across multiple configurations and tasks. The Koopman embedding $\Psi(x)$ is offline trained only once on a fixed configuration, and the policy gain matrix K is online updated for multi-configuration adaptation via model-free RL in the Koopman embedding space. The symbol “-” and “[|]” denote serial and parallel connections between segments, respectively. **b**, Type I elephant-trunk robots: Each segment features a honeycomb structure with four pneumatic actuators. Honeycomb segments’ properties vary with the hole width between walls (superscript indices $1, \dots, N$ denote the stiffness levels determined by the honeycomb holes detailed in Supplementary Fig. 5 and Supplementary Table 1) and segment lengths (subscript indices ‘S, M, L’ represent segment lengths: Short, Medium, and Long). The centerline lengths for short and long honeycomb segments are 65.5 mm and 101.5 mm, respectively. **c**, Type II soft-muscle robots: Each segment is composed of three parallel pneumatic muscles, each driven by a pneumatic actuator. The centerline lengths of the short, medium, and long muscle segments are 159 mm, 194 mm, and 229 mm, respectively. **d**, Type III worm robots: Each segment features a worm structure with three pneumatic actuators. The short, medium, and long worm segments have centerline lengths of 190 mm, 220 mm, and 270 mm, respectively.

Reviewer Comment

3. The criteria for selecting the 30 configurations that are used do not seem to be discussed. Please provide some context or rationale behind these choices that lead to these specific configurations.

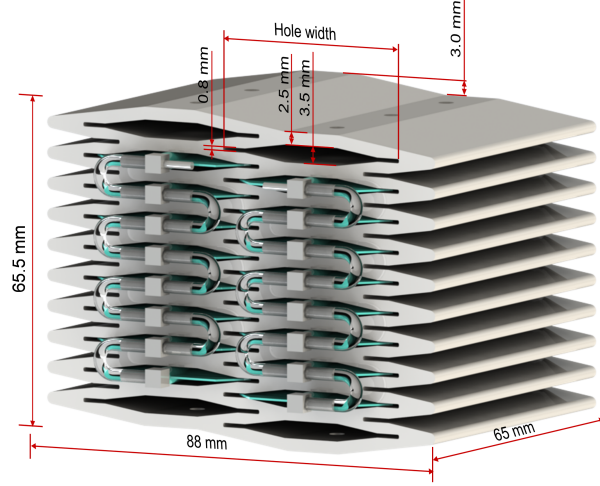


Fig. 9: Schematic diagram of the honeycomb segment structure. The stiffness level of the honeycomb segment can be adjusted by modifying the hole width.

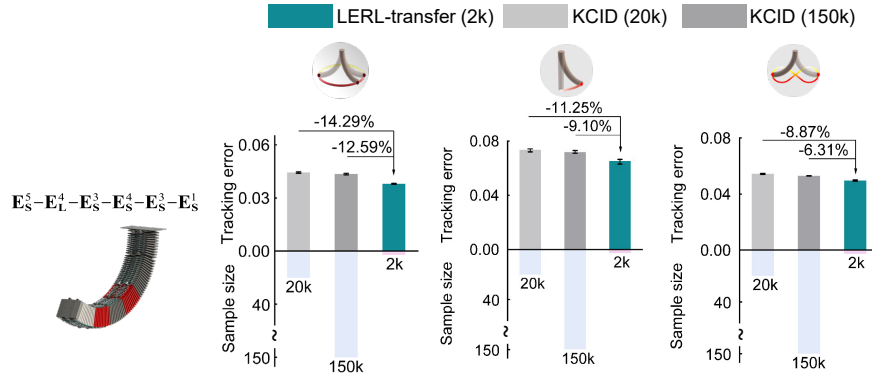


Fig. 10: Performance validation of LERL on a Type I elephant-trunk robot configuration with six segments. LERL reduces tracking errors by 12.59%, 9.10%, and 6.31% in target reaching, circular, and bow tie tracking tasks compared to KCID with 150k samples, similar to the results obtained for configurations with two, three, and four segments.

Response

We selected 30 robot configurations (expanded to 33 in the revised manuscript) because they induce distinct and diverse system dynamics, which are sufficient and convincing for validating the transferability of our method. Specifically, these configurations span robots with varying numbers of segments (from a single segment to six segments), three-segment robots with different assembly orders, as well as hybrid-assembled robots composed of different segment types and hybrid parallel-series configurations. Together, these configurations form a robot configuration library that captures a wide range of heterogeneous and nonlinear dynamics and supports diverse complex tasks. For instance, high-stiffness honeycomb segments can be assembled to construct soft robots capable of performing tasks such as hammering nails and serving drinks, which require stable operation under high payloads. In contrast, more flexible worm-like segments can be assembled to form soft robots suited for quick hand-eye reaction games, which demand instantaneous high-speed responses.

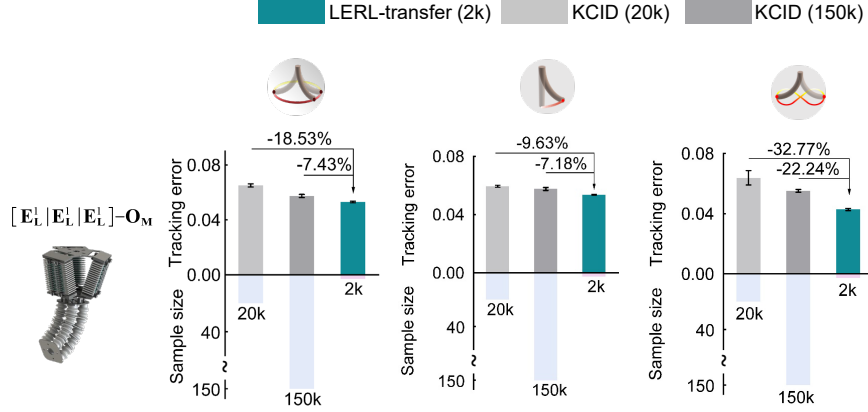


Fig. 11: Performance validation of LERL on a new parallel-series hybrid robot configuration. LERL reduces tracking errors by 7.43%, 7.18%, and 22.24% in circular tracking, bow tie tracking, and target reaching tasks compared to KCID with 150k samples, achieving performance consistent with that observed on previously studied configurations.

The above statements have been included on Page 12 of the revised manuscript and listed below.

“Our three soft robotic platforms encompass 33 distinct configurations (Fig. 8), including diverse assemblies of honeycomb, muscle, and worm segments with varying segment numbers, assembly orders, cross-type combinations, and hybrid parallel-series structures. Collectively, these configurations constitute a representative robot configuration library that spans a wide range of heterogeneous and nonlinear dynamics and supports diverse complex tasks. For example, high-stiffness honeycomb segments enable stable, high-payload tasks such as hammering nails and serving drinks, whereas flexible worm-like segments facilitate quick hand-eye reaction tasks requiring instantaneous high-speed motions (Fig. 5).”

In addition, in the revised manuscript, three new configurations were added for validation. First, a six-segment configuration was used to show that LERL scales to higher-dimensional soft robots (see Fig. 10 above and Fig. 4a, Figs. 11-12 of the revised manuscript). Second, we designed a heterogeneous parallel-series hybrid robot (see Fig. 11 above and Fig. 4a, Figs. 11-12 of the revised manuscript) to validate that LERL can handle both internal coupling effects and external disturbances in a more complex configuration. Third, we added a configuration composed of three different segment types to further evaluate policy transfer across diverse robot morphologies (see Fig. 12 below and Fig. 4a, Figs. 11-12 of the revised manuscript).

Reviewer Comment

4. Koopman control method with inertial dynamics (KCID) is defined in the Results section but is referenced prior to it in the caption of Table 1.

Response

We have provided the full name of KCID, i.e., Koopman control method with inertial dynamics, in the caption of Table 1, according to your comment. We also checked the entire manuscript to ensure that all abbreviations are defined before usage.

The following content is included in the caption of Table 1 in the revised manuscript (Page 4).

“Additionally, Koopman control method with inertial dynamics (KCID) with time-delay (TD) embeddings⁷ requires 18k samples per configuration,...”

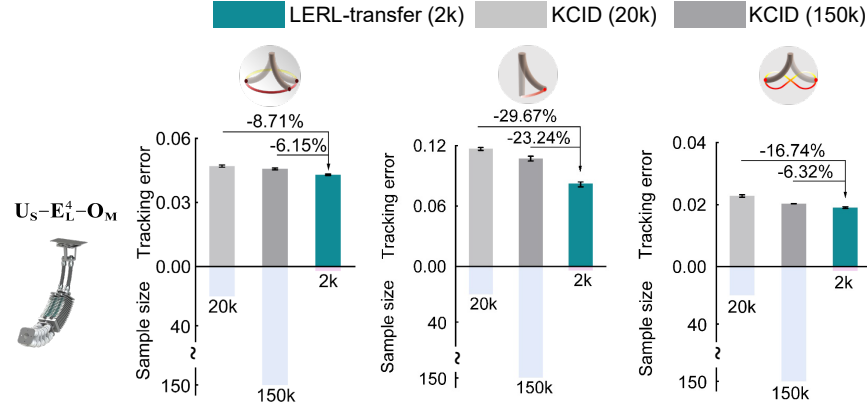


Fig. 12: Performance validation of LERL on a configuration assembled by three different segment types. LERL reduces tracking errors by 6.15%, 23.24%, and 6.32% for circular tracking, bow tie tracking, and target reaching tasks compared to KCID with 150k samples.

Reviewer Comment

5. *KCID seems to be referred to as KLQR (which comes from the author of Ref. 7) in Fig. 9 and Fig. 10 but not their captions. Please maintain consistency.*

Response

Thank you so much for your careful review. We have corrected this error in the revised manuscript.

Reviewer Comment

6. *Consider adjusting line thickness in figures such as Fig. 3c and Fig. 6e to improve readability.*

Response

Thank you for the suggestion to improve the quality of the figures. In the revised manuscript, the line thickness in Fig. 3c of the revised manuscript has been increased from 3 pt to 5 pt, and the line thickness in Fig. 6e of the revised manuscript has been increased from 4 pt to 7 pt. In addition, we have enlarged the font sizes in Figs. 2, 4, 11-12 of the revised manuscript to improve clarity and readability.

Reviewer Comment

7. *The naming convention for the muscle and worm types are identical, though the accompanying images of the actual assemblies do clarify which type is used. For more clarity, consider modifying the notation, perhaps including an additional subscript denoting which type is used.*

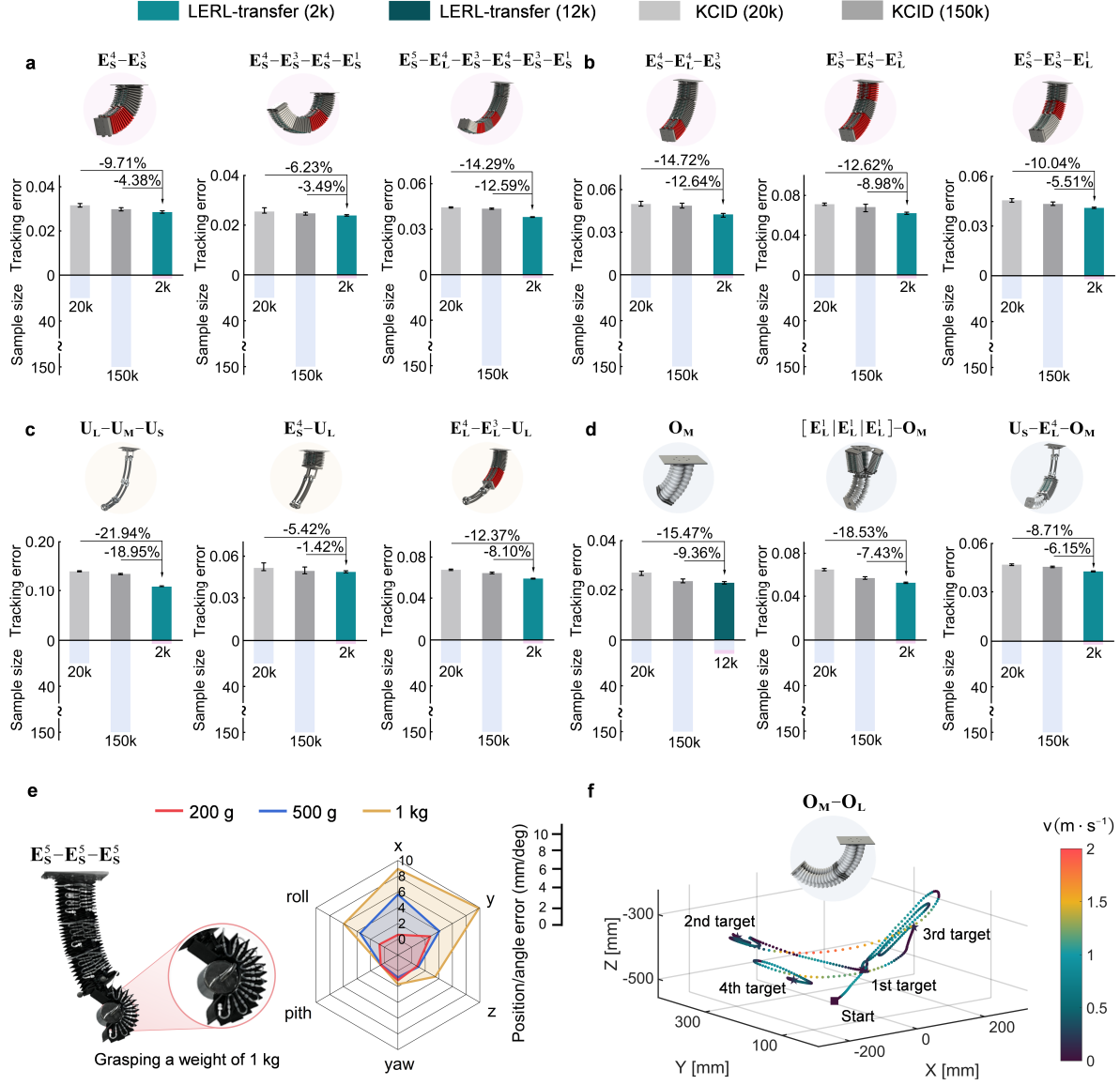


Fig. 13: Policy adaptation performance to multiple configurations of three soft robot prototypes.

Response

According to the comment, we now use the symbol ‘E’ to only represent the segment of the elephant-trunk robot, introduce the symbol ‘U’ to represent the segment of the muscle robots, and use the symbol ‘O’ to represent the segment of worm soft robots. We have also replotted Figs. 4, 6-10, and updated the related descriptions to reflect this new naming convention, improving clarity and consistency throughout the manuscript. For convenience, we provide Fig. 13 above (Fig. 4 in the revised manuscript) as an illustrative example.

Response To Reviewer #3

Overall Comments

1. This work presents an approach for soft robotics control through the use of reinforcement learning in a Koopman embedding space. They claim a 75x reduction in training samples needed compared to state-of-the-art methods without a significant impact on control optimization. Further, it is claimed that this model is applicable to diverse soft robotic morphologies, including variations in materials, stiffnesses, structural geometry, and assemblies.

Response

We sincerely appreciate the time and effort you devoted to reviewing our work, which has been instrumental in improving the quality of the manuscript. The major revisions in response to your comments are summarized as follows.

- We have added comparisons with the recommended reference works to better highlight the novelty of our approach.
 - We have conducted additional comparative experiments to evaluate the performance of LERL under varied amounts of training data.
 - We designed three new robot configurations, including a parallel-series hybrid configuration, to demonstrate the policy transfer capability of LERL across heterogeneous soft robots with distinct morphologies.
-

Reviewer Comment

2. The significance of this work does not rely on the novel use of Koopman operators or reinforcement learning in soft robotics. Its novelty is in the marriage between the two and the use of a Koopman embedding space to streamline and generalize learned control methods. It is this reviewers belief that this combination is a currently under-explored method for generating control policies for soft robotics. However, other work in similar conceptual spaces (reinforcement learning married to Koopman embedded space for the control of soft/continuum robots) does exist [R1-R3]. The novelty of this work seems to be primarily in the validation tasks (hammering, serving drinks, calligraphy, etc) using this control paradigm. As such, additional data and analysis of these tasks may be helpful to better support this work.

References:

- [R1] Sinha, Anirban, and Yue Wang. Koopman operator-based knowledge-guided reinforcement learning for safe human-robot interaction. *Frontiers in Robotics and AI* 9 (2022): 779194.
- [R2] Ji, Guanglin, et al. Efficient real2sim2real of continuum robots using deep reinforcement learning with Koopman operator. *IEEE Transactions on Industrial Electronics* (2025).
- [R3] Abraham, Ian, and Todd D. Murphey. Active learning of dynamics for data-driven control using Koopman operators. *IEEE Transactions on Robotics* 35.5 (2019): 1071-1083.

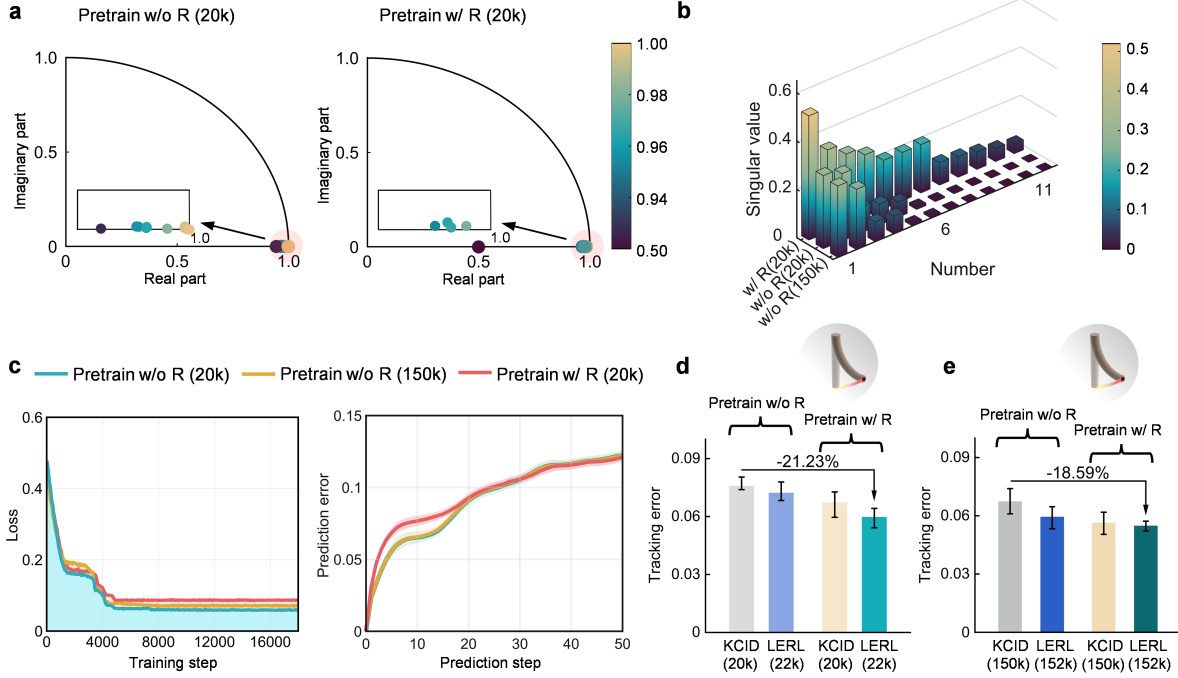


Fig. 14: Offline Koopman embedding training and online policy learning for honeycomb segment E_S^1 . **a**, The open-loop eigenvalue plots for Koopman models pretrained with (w/) or without (w/o) regularization (R). The eigenvalues pretrained w/ regularization locate farther from the unit circle, resulting in a larger stability margin; 20k=20000. **b**, The singular value distribution (SVD) of the controllability matrix of Koopman models. Pretraining w/ regularization leads to SVDs with higher magnitudes. **c**, The training loss (equation (5) in Methods) and prediction error of the Koopman model, Prediction error $:= \sqrt{\frac{1}{L} \sum_{l=1}^L \|x_l - \hat{x}_l\|^2}$, L denotes the amount of samples. **d-e**, Performance comparison of KCID and LERL during pretraining w/ or w/o regularization, Tracking error $:= \frac{1}{M \cdot N} \sum_{i=1}^M \sum_{k=1}^N \|\bar{x}_{i,k} - \bar{x}_{r,i,k}\|^2$, $\bar{x}, \bar{x}_r \in \mathbb{R}^{12}$ are the normalized real state variable (pose and velocity) and its reference, N is the algorithm horizon, and $M = 5$ is the number of experiments.

Response

Thanks for your insightful comment. We address your concerns point by point below.

Novelty to Koopman-based deep reinforcement learning (DRL). We agree that combining Koopman operators with reinforcement learning is a promising direction for soft robotics. Our proposed LERL enables online model-free policy learning directly on real robots in a linear Koopman embedding space, which is fundamentally different from the DRL approaches in [R1] and [R2] recommended in the comment (corresponding to references [42] and [43] in the revised manuscript). **Those methods identify a Koopman model primarily to serve as a simulator or predictor, while policy learning is still conducted in the original state space.** Specifically, these DRL approaches rely on high-fidelity simulators to generate training data and may suffer from the sim-to-real gap, even when the learned Koopman model exhibits small prediction errors. As shown in this work, a Koopman model with good prediction accuracy may still lack essential control properties such as controllability or stabilizability (see Figs. 14a-e above and Fig. 3a-e in the revised manuscript), because conventional Koopman modelling typically optimizes only regression-based objectives.

In contrast, our method incorporates control-oriented regularization alongside regression objectives to

explicitly enforce key open-loop control properties (e.g., stability and controllability) in the embedding space, which are critical for closed-loop performance. As shown in Figs. 14a-e above (Figs. 3a-e of the revised manuscript), this control-oriented regularization is essential for achieving superior control performance compared to regression-only Koopman modelling, thereby bridging the gap between data-driven modelling and closed-loop control.

More importantly, our approach enables **online model-free policy learning directly in the linear embedding space**. This allows policy updates within 10 ms (Table 5 below and Table 2 of the revised manuscript), smaller than the control sampling interval 20 ms, and enables rapid adaptation to dynamic variations across robot configurations. Hence, this design addresses the sim-to-real gap that limits existing Koopman-based DRL approaches due to our real-time direct policy update mechanism, making LERL particularly well-suited for real-world soft robotic systems, especially those exhibiting significant dynamics variations, such as the 33 robot configurations considered in our work.

Table 5: 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

Novelty to model-based controller. In view of the above statements, LERL is also fundamentally different from the model-based approach in [R3] mentioned in the comment (reference [29] in the revised manuscript), which learns a Koopman model for model-based controller design. In such settings, control performance **can be limited by modelling inaccuracies and by the potential lack of stabilizability, as regression-based Koopman learning does not explicitly enforce control properties (e.g., stability and controllability).**

In the revised manuscript, we have cited the suggested works [R1]-[R3] and include the following content to highlight the novelty of our work to DRL and model-based control methods.

“The Koopman operator^{27–29} offers a promising alternative by capturing nonlinear dynamics through linear evolutions in an infinite-dimensional embedding space.....Despite these advances, the aforementioned approaches require repetitive modelling and controller design to adapt to new configurations, limiting their practical applicability in reconfigurable robotic systems.”

“A common approach is to design physical simulators as an alternative to real-world training, including the use of Koopman operators to learn surrogate models for data generation^{42,43}. However, RL algorithms such as deep deterministic policy gradient⁴⁴ and proximal policy optimization⁴⁵ still require millions of data samples to converge³⁹. Moreover, building high-fidelity simulators of soft robots is challenging due to their nonlinear, configuration-dependent dynamics, exacerbating the sim-to-real gap.”

More data and analysis in tasks. Beyond the novelty of our methodology, LERL empowers robots performing previously unattainable real-world skills, including carpenter-inspired nail hammering into wooden boards, bartender-like drinks serving with safe human-robot interaction, fluid calligraphy brush handwriting, and quick hand-eye reaction games mimicking athletic and child training. According to your comment, we have also added more data and analysis regarding the challenging tasks. For instance, in the hammering nail

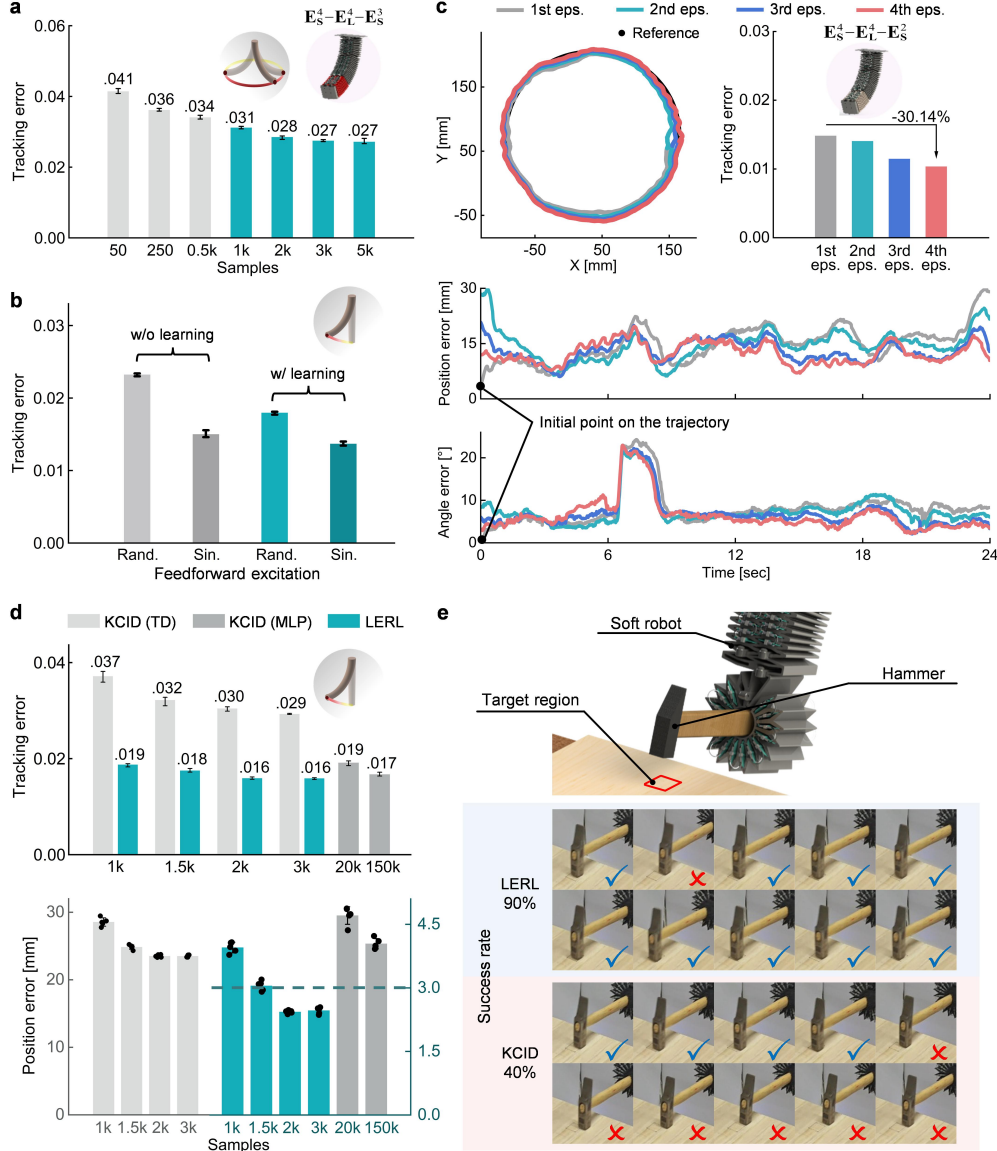


Fig. 15: Auxiliary performance evaluations of dataset properties and success rate. **a**, Transfer performance under different dataset sizes and compositions. **b**, Task performance under feedforward policies trained with datasets generated using sinusoidal and random input excitations. **c**, Four-episode (eps.) learning performance in circular tracking of one elephant-trunk robot. **d**, Transfer performance comparison under varied amounts of training data. **e**, Success rate comparison for hammering a fixed point within a predefined positional tolerance of 3 mm.

task, we have compared the success rate of hammering a fixed point within a positional tolerance of 3 mm (refer to the reply for your 3rd comment) against KCID. The results demonstrate that LERL succeeds in 9 out of 10 trials, whereas KCID succeeds in only 4 out of 10 trials (see Fig. 15e above and Fig. 9e in the revised version).

The accompanying discussion on Page 10 in the revised manuscript is listed below.

“We further compared LERL with KCID in terms of precise control capability by evaluating their performance in hammering a fixed point within a positional tolerance of 3 mm (Fig. 9e). LERL succeeds in

9 out of 10 trials, whereas KCID succeeds in only 4 out of 10 trials.”

In the serving drinks task that requires smooth operations, we have added the following content (Page 10 in the revised manuscript) to report the average velocity and its variance during execution. These results indicate that the worm robot operates smoothly during the human-robot interaction task.

“The average end-effector positional velocity is $0.018 \text{ m} \cdot \text{s}^{-1}$ with a variance of $0.000695 \text{ m}^2 \cdot \text{s}^{-2}$, demonstrating smooth motions throughout the serving process.”

Reviewer Comment

3. One of the main claims of the work is that there is a 75x reduction in training samples required compared to other methods. How does the learned control policy compare to state-of-the-art control policies on these tasks when given varied amounts of training data? What is the failure rate, if any, of the robot when performing these tasks? It seems that the claim of 75x less training data required is a major part of this work, however comparisons to other methods and validations of this claim seem to be misrepresentative. From the work itself: “LERL using 22k training samples, 20k for offline Koopman embedding training and 2k for policy generation.” Indicating that 20k samples are used for training the Koopman embedding space and an additional 2k are used to train the control policy. In table 1, the learning methods used in this work are compared directly to other learning methods, however, the compared learning methods have all of their training data samples bundled together as opposed to just the policy generation samples (at least as far as I can tell from reading the cited works). This drastically under-represents the samples required for this work and when taken into account puts it in line with many state-of-the-art methods for control policy generation. I suggest reviewing table 1 and putting in a disclaimer as to why only the policy generation training data is accounted for in this work as opposed to all training data required for this combined learning and Koopman operator approach. Further, I would go over the table again and ensure it is a fair comparison, especially with the other methods using Koopman operators (make sure that you are not counting their embedded space training data if you are not including your own). With this view on training data, the performance of this model is still greater than many state-of-the-art methods, however it requires substantially more training data than is eluded to in the claims and tables of this work.

Response

Thank you so much for your constructive comment. We address your concerns point by point below.

-How does the learned control policy compare to state-of-the-art control policies on these tasks when given varied amounts of training data?

According to your comment, we have now compared LERL with the state-of-the-art (SOTA) KCID method (published in Science Robotics, November 2023, and reported to achieve the best performance with inertial dynamics among soft robots) under varying data sizes. LERL was implemented with the multi-layer perceptron (MLP) embedding, while KCID was implemented using both the time-delay (TD) embedding and MLP embedding, termed KCID-TD and KCID-MLP, respectively. LERL and KCID-TD were evaluated using datasets of 1k, 1.5k, 2k, 3k, and 5k samples, where both methods used 500 samples to train the feedforward policy. The remaining samples were used for online learning in LERL and for updating model parameters in KCID-TD, respectively. For KCID-MLP, we used 20k and 150k samples to derive the model-based control policy for comparison. As shown in Fig. 15d above (corresponding to Fig. 9d in the revised manuscript), LERL achieves substantially lower tracking error than the KCID-TD under the same sample condition. For instance, with 2k samples, LERL reduces tracking error by 46.7 % compared to KCID-TD.

In addition, LERL outperforms KCID-MLP even when the latter uses substantially larger datasets (20k and 150k samples).

The corresponding statements are included on Page 18 of the revised manuscript and also listed below.

“We compared LERL with KCID under varying data sizes. The LERL was implemented using the transferred policy with MLP embeddings, while KCID was implemented using both TD embeddings and MLP embeddings. LERL and KCID with TD embeddings were evaluated using datasets of 1k, 1.5k, 2k, 3k, and 5k samples, where both methods used 500 samples to train the feedforward policy. The remaining samples were used for online learning in LERL and for updating model parameters in KCID with TD embeddings, respectively. In KCID with MLP embeddings, we used 20k and 150k samples to derive the model-based control policy for comparison. As shown in Fig. 9d, LERL achieves substantially lower tracking error than KCID with TD embeddings under the same sample condition, and also outperforms KCID with MLP embeddings even when the latter uses substantially larger datasets (20k and 150k samples).”

-What is the failure rate, if any, of the robot when performing these tasks?

We have added additional experiments on target reaching and hammering nail tasks to quantify the failure rate. First, for the target reaching task, we follow the criteria in [1] (corresponding to reference [45] in the revised manuscript), where a task is considered successful if the end-effector position error is less than 1% of the arm length. Given the length of the chosen configuration $E_S^5 - E_S^5 - E_S^5$ is 300 mm, thereby an episode is considered successful if this final error is less than 3 mm; otherwise, it fails. Each experiment setting was independently repeated 5 times. According to the results reported in Fig. 15d above (corresponding to Fig. 9d in the revised manuscript), the success rate of LERL is 0% with 1k samples, 60% with 1.5k samples, and 100% with 2k samples. However, KCID fails under all the above cases.

Second, we have conducted comparative experiments for LERL and KCID under the 2k sample condition on the hammering nail task, where a trial is considered successful if the hitting point falls within the target region, which is 3 mm expanded beyond the head of the hammer. **LERL achieves a success rate of 9 out of 10 trials, whereas KCID succeeds in only 4 out of 10 trials** (see Fig. 15e above and Fig. 9e in the revised manuscript).

The corresponding statements are included on Page 18 and also listed below.

“To further evaluate the practical utility of our method, we adopt a success criterion requiring the position tracking error to be within 1% of the robot length⁴⁵, i.e., 3 mm of 300 mm. Under this criterion, LERL reaches a 100% of success rate when the number of samples used for policy transfer exceeds 2k (Fig. 9d). In contrast, KCID fails in reaching a precision of 3 mm under all the adopted sample scenarios. With the same robot configuration, we further conducted a comparative study between LERL (transferred with 2k samples) and KCID (trained with 2k samples) to complete hammering experiments with a positional tolerance of 3 mm. The results demonstrate that LERL achieves a success rate of 9 out of 10 trials, whereas KCID succeeds in only 4 out of 10 trials (Fig. 9e, Supplementary Video 1).”

- It seems that the claim of 75x less training data required is a major part of this work, however comparisons to other methods and validations of this claim seem to be misrepresentative. From the work itself: “LERL using 22k training samples, 20k for offline Koopman embedding training and 2k for policy generation.” Indicating that 20k samples are used for training the Koopman embedding space and an additional 2k are used to train the control policy. In table 1, the learning methods used in this work are compared directly to other learning methods, however, the compared learning methods have all of their training data samples bundled together as opposed to just the policy generation samples (at least as far as I can tell from reading the cited works). This drastically under-represents the samples required for this work and when taken into account puts it in line with many state-of-the-art methods for control policy generation. I suggest reviewing

table 1 and putting in a disclaimer as to why only the policy generation training data is accounted for in this work as opposed to all training data required for this combined learning and Koopman operator approach. Further, I would go over the table again and ensure it is a fair comparison, especially with the other methods using Koopman operators (make sure that you are not counting their embedded space training data if you are not including your own). With this view on training data, the performance of this model is still greater than many state-of-the-art methods, however it requires substantially more training data than is eluded to in the claims and tables of this work.

Thank you for this constructive comment. Although 22k data samples (20k for Koopman embedding training and 2k for policy learning) are used on a single segment configuration E_S^1 , and only 2k data samples are used for policy transfer to diverse robot configurations, we agree that it would be unfair to report only the 2k samples while neglecting the 20k samples required for training the Koopman embedding.

Accordingly, in the revised manuscript, we explicitly distinguish between two scenarios: (i) Koopman embedding training and policy learning with 22 samples on a fixed configuration and (ii) policy transfer across diverse configurations with 2k samples. The corresponding data requirements for each scenario are now separately reported in Table 6 below (Table 1 of the revised manuscript). Additionally, we clarify the use of an additional 10k data samples when transferring from the honeycomb segment to the worm robot, where these samples are employed for an initial policy initialization procedure.

Following your comment, we now claim the data efficiency of LERL separately for the single-segment training scenario and for transfer to other configurations. As shown in Fig. 3f of the revised manuscript, LERL trained with 22k samples outperforms KCID trained with 150k samples, achieving a $6.8\times$ reduction in required data. In the policy transfer setting, LERL requires only 2k samples to outperform KCID trained with 150k samples (used to train a configuration-specific Koopman model), reducing the data samples up to $75\times$.

In summary, we have made the following modifications in the revised manuscript to strengthen the rigour of our claims. (i) We have revised Table 1 to explicitly distinguish between the data samples required for Koopman embedding training and those required for policy transfer. (ii) We have updated both the abstract and the main text to clarify that LERL reduces the required data by $6.8\times$ in the single-segment training scenario and by up to $75\times$ when transferring to other robot configurations.

Reviewer Comment

4. Another claim is that the control policies work across 30 soft robot configurations, however the notability of these configurations is somewhat suspect. There are three main soft robotic assemblies used to validate the control policy, all three are continuum robots utilizing pneumatic actuation systems. The primary variation in the robots themselves is their stiffness and the amount of connected segments or blocks they use (which can be constructed of varying assemblies). Overall, these configurations all have relatively similar movements and the control methods used are very similar. I think this claim may need to be reviewed as it seems to over-inflate the applicability of the learned policy when it is being applied to functionally very similar mechanisms.

Response

Thank you for your comments. We agree that the adopted configurations exhibit relatively similar motion patterns while possessing distinct underlying dynamics. Our method is built to transfer across soft robot configurations with different dynamics, provided that they share some common fundamental dynamical

Table 6: Comparison with state-of-the-art (SOTA) methods for control of soft robots. Our LERL framework enables policy transfer across 33 distinct configurations, whereas most SOTA methods are limited to a fixed configuration. LERL requires 22k samples to generate a policy for a fixed configuration E_s^1 , including 20k samples for training the Koopman embedding and 2k samples for online policy updates during control execution. Policy transfer across robot configurations requires only 2k samples. This sample demand, particularly in policy transfer, is substantially fewer compared to RL methods such as proximal policy optimization (PPO)⁴⁵ ($\geq 5250k$, $2600\times$ more), trust region policy optimization (TRPO)⁴⁶ ($\geq 300k$, $150\times$ more), and deep deterministic policy gradient (DDPG)⁴⁴ ($\geq 100k$, $50\times$ more). Additionally, Koopman control method with inertial dynamics (KCID) with time-delay (TD) embeddings⁷ requires 18k samples per configuration, which is $9\times$ more than LERL with MLP embeddings in policy transfer. When both using the same MLP embeddings, LERL reduces the number of training samples by $6.8\times$ for a fixed configuration and by $75\times$ for policy transfer across robot configurations. Despite using the fewest samples, LERL outperforms all other SOTA methods in terms of speed, acceleration, and payload capacity. LSTM: Long short-term memory; RNN: Recurrent neural network.

Work	Method	Config.	Config. transfer	Robot type	Hybrid assembly	Online learning	Sample Fixed config.	Sample Transfer	Speed ($m \cdot s^{-1}$)	Accel ($m \cdot s^{-2}$)	Max Load (g)
Ours	LERL	33	32	3	✓	✓	22k (MLP)	2k	Type I: 0.78	11.9	1000
									Type III: 1.89	22.34	200
Ref ⁴⁶	LSTM + TRPO	1	-	1	✗	✗	$\geq 300k$		0.17	-	165
Ref ⁴¹	RNN + RL	1	-	1	✗	✗	28k		-	-	105
Ref ⁴⁵	LSTM+PPO	1	-	1	✗	✓	$\geq 5250k$		0.3	-	-
Ref ⁴⁴	DDPG	1	-	1	✗	✗	$\geq 100k$		-	-	20
Ref ⁴⁷	Q-learning	1	-	1	✗	✓	$\geq 38k$		-	-	500
Ref ⁷	KCID	2	-	1	✗	✗	18k (TD)		1.52	11.6	-
							150k (MLP)				
Ref ³⁴	Koopman + MPC	1	-	1	✗	✗	247k		-	-	-
Ref ³²	Koopman + MPC	1	-	1	✗	✗	$\geq 43k$		-	-	-
Ref ³¹	Koopman + MPC	1	-	1	✗	✗	$\geq 325k$		-	-	275
Ref ²³	Hybrid model + PID	1	-	1	✗	✗	120k		0.08	0.07	-

characteristics. In response to your comment, we have assembled three additional robot configurations with more pronounced dynamic diversity. First, we introduced a parallel-series hybrid robot (see Fig. 16 below), which exhibits substantial dynamic variations compared to the previously studied series-only configurations. Second, we constructed a hybrid robot composed of all three segment types (see Fig. 17 below), resulting in significantly different dynamics due to the heterogeneous structures, materials, stiffness levels, and actuation mechanisms of the segments (see Table 7 below). Third, we evaluated the performance on an elephant-trunk-shaped robot with six honeycomb segments (see Fig. 18 below), extending beyond the four-segment configurations considered in the previous experiments.

In all the newly added experiments, LERL with only 2k transfer samples achieves substantial reductions in tracking error compared to KCID trained with 150k samples in the target reaching, circular, and bow tie tracking tasks (see Figs. 16-19 below). Please refer to Figs. 11-12 of the revised manuscript and the associated discussions for details. These results demonstrate the potential of LERL to scale to parallel robots with octopus-like architectures, heterogeneous robots composed of different materials and structural designs, and soft robots with a larger number of segments. In general, the additional experiments and the corresponding results highlight the practical value of our approach for controlling more complex soft robotic systems with diverse morphologies.

Table 7: Comparison among three types of segments.

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

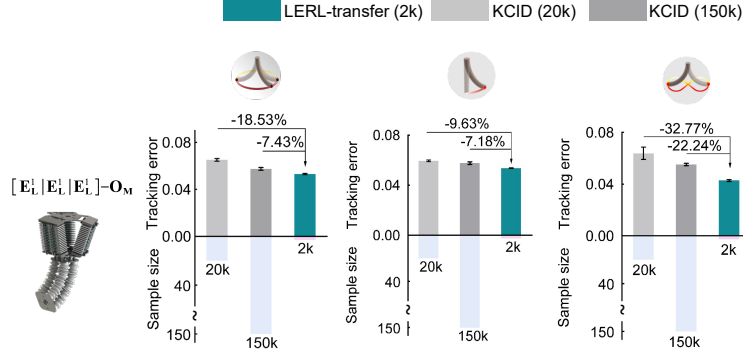


Fig. 16: Performance validation of LERL on a new parallel-series hybrid robot configuration. LERL reduces tracking errors by 7.43%, 7.18%, and 22.24% for circular tracking, bow tie tracking, and target reaching tasks compared to KCID with 150k samples, achieving performance consistent with that observed on previously studied configurations.

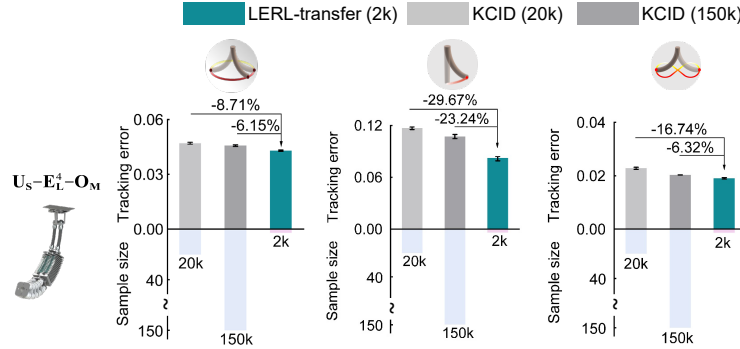


Fig. 17: Performance validation of LERL on a configuration assembled by three different segment types. LERL reduces tracking errors by 6.15%, 23.24%, and 6.32% for circular tracking, bow tie tracking, and target reaching tasks compared to KCID with 150k samples.

We acknowledge that the 33 configurations considered in this work do not encompass all types of soft robots encountered in real-world applications, such as those with more complex parallel structures and diverse actuation mechanisms. In the Discussion section of the revised version, we have stated that our future work will extend the LERL method to even more complex and diverse robotic architectures, including unified and adaptable control of both rigid and soft robots. Notably, we have already applied the LERL policy to control a rigid-bodied robot arm in a simulated experiment (see Fig. 19 below, and Supplementary Section 6 of the revised manuscript), demonstrating the potential of our approach for transferability across both soft- and rigid-bodied systems.

The statements associated with the newly added experiments and the discussions are provided on Page 9

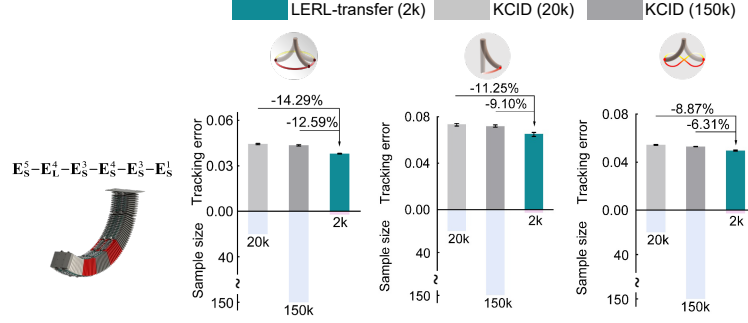


Fig. 18: Performance validation of LERL on a Type I elephant-trunk robot configuration with 6 segments. LERL reduces tracking errors by 12.59%, 9.1%, and 6.31% in target reaching, circular, and bow tie tracking tasks compared to KCID with 150k samples, similar to the results obtained for configurations with two, three, and four segments.

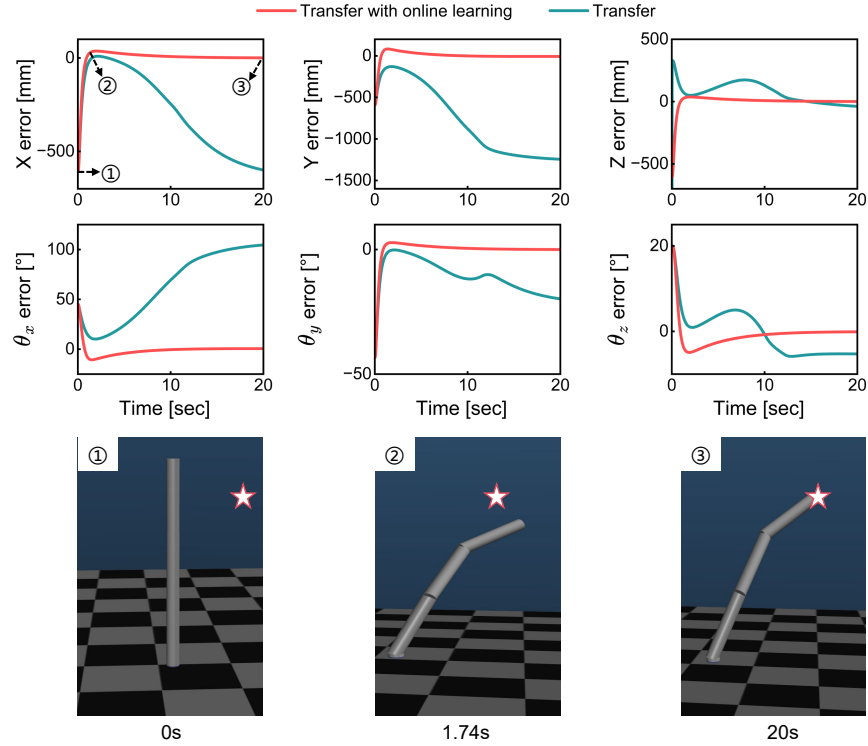


Fig. 19: Policy adaptation to rigid robots. Pose error curves for policy transfer with or without online learning scenarios, along with the pose of the rigid-body robot at selected moments. The highlighted star is the reference target of the robot tip.

and Page 11 respectively, which are summarized below.

On Page 11: “Moreover, with only 2k samples, LERL enables rapid policy transfer to heterogeneous hybrid-assembled robots, including a parallel–serial hybrid soft robot (Fig. 4d).”

“The control policy initially trained in E_S^1 was efficiently transferred using a padding procedure detailed in Supplementary Section 5, and refined online to adapt to new configurations comprising two, three, and six segments. As shown in Fig. 4a, LERL-transfer with 2k samples achieves lower tracking errors in all

scenarios compared to KCID with 150k samples, reducing the sample size by 75× in policy transfer across robot configurations.”

In the Discussion section on Page 11: “Future work will extend the LERL method to robotic systems with more diverse and complex architectures. This includes unified and adaptable control of both rigid and soft robots (see Supplementary Section 6 for an elementary validation of rigid robots).”

Reviewer Comment

5. The methodology is sound and the results seem to be genuine. There also appears to be enough data and detail provided for the methods of this work to be reproduced.

Response

Thank you so much for your positive evaluation of our work.

Reviewer Comment

6. Additionally, many figures in this work are unclear and cluttered with text size dropping substantially below the default font size used in this work. General cleanup and reformatting of the figures for clarity may significantly enhance this work.

Response

Thank you for this helpful comment. According to the suggestion, we have comprehensively reformatted the figures to reduce visual clutter and improve legibility. Specifically, we increased the font sizes of all axis labels, legends, and annotations to match the manuscript’s default font size (see the updated Fig. 2, Fig. 4, Fig. 7, Figs. 11-12 in the revised manuscript). We also adjusted line widths and marker sizes to enhance visibility (see Fig. 3c and Fig. 6e in the revised manuscript). In addition, we reorganised the multi-panel layouts to reduce overcrowding and improve spacing (see Fig. 8 in the revised manuscript).

<https://www.overleaf.com/project>

Reviewer Comment

7. This reviewer recommends revision and additional clarity into the comparison methods used to justify claims made in this work.

Response

Thank you for the constructive suggestion. As discussed previously, we have made the following revisions to make the comparison more rigorous and transparent. First, we have added comparisons with the recommended reference works to better highlight the novelty of our approach. Second, we have conducted additional comparative experiments to evaluate the performance of LERL under varied amounts of training data. Third, we designed three new robot configurations, including a parallel-series hybrid configuration, to demonstrate the policy transfer capability of LERL across heterogeneous soft robots with distinct morphologies.

References

- [1] Muhammad Sunny Nazeer, Cecilia Laschi, and Egidio Falotico. RL-based adaptive controller for high precision reaching in a soft robot arm. *IEEE Transactions on Robotics*, 40:2498–2512, 2024.

Response letter

Reviewer 1's comment:

The authors have a clearly organized response to all of the reviewers, which is appreciated. I only have a few minor comments lingering.

- Fig. 4d second y axis is inside of subfigure 4e.

Response: Thank you for this valuable comment. In the revised manuscript, we have adjusted the size and layout of Fig. 8d (corresponding to Fig. 4 in the previous response) to ensure that all elements of Fig. 8d are properly positioned and no longer overlap with or appear inside Fig. 8e.

- 4a not clear what is gray vs. teal, unless we are supposed to assume the color code in d, which doesn't make much sense. Based on the description, I understand it to be gray is varying the feedforward training samples, and teal is varying the transfer sample size.

Response: Thank you for this helpful comment. In the revised manuscript, we have added explicit descriptions in Fig. 8a (corresponding to Fig. 4 in the previous response) to clearly distinguish the two conditions. The gray bars now explicitly indicate the performance without online learning (i.e., using only feedforward training samples), while the teal bars represent the performance with online learning. This clarification removes ambiguity regarding the color coding.

- I did review the Git Repository and it seems well documented.

Response: Thanks a lot for your careful review.

Reviewer 2's comment:

- The authors have addressed my comments properly. I don't have any other comments and I can recommend the publication of the paper.

Response: Thank you so much for your recommendation.

Reviewer 3's comment:

- The authors have carefully addressed all of my previous comments. The additional analyses and clarifications have strengthened the manuscript, and I am satisfied with the revisions. The additional robot configurations and enhanced clarity of figures add substantially to this work. Additionally, the improved clarity on how LERL is compared to other methods is appreciated. It appears that all comparisons are direct and do not misrepresent critical metrics. I have no further comments and recommend acceptance in its current form.

Response: Thank you so much for your recommendation.