Journal of Mechanisms and Robotics. Received February 06, 2024;
Accepted manuscript posted February 7, 2025. doi:10.1115/1.4067872

Learning-Based Position and Orientation Control of a Hybrid Rigid-Soft Arm Manipulator

Kendall Koe *
Computer Science
University of Illinois
Urbana, Illinois 61801
Email: kfkoe2@illinois.edu

Samhita Marri *
Electrical and Computer Engineering
University of Illinois
Urbana, Illinois 61801
Email: marri2@illinois.edu

Benjamin Walt
Mechanical Science and Engineering
University of Illinois
Urbana, Illinois 61801
Email: walt@illinois.edu

Shivani Kamtikar
Computer Science
University of Illinois
Urbana, Illinois 61801
Email: skk7@illinois.edu

Naveen Kumar Uppalapati
National Center for Supercomputing Applications
University of Illinois
Urbana, Illinois 61801
Email: uppalap2@illinois.edu

Girish Krishnan
Industrial and Systems Engineering
University of Illinois
Urbana, Illinois 61801
Email: gkrishna@illinois.edu

Girish Chowdhary
Computer Science
University of Illinois
Urbana, Illinois 61801
Email: girishc@illinois.edu

1 Abstract

We present a position and orientation controller for a hybrid rigid-soft manipulator arm where the soft arm is extruded from a two degrees-of-freedom rigid link. Our approach involves learning the dynamics of the hybrid arm operating at 4 Hz and leveraging it to generate optimal trajectories that serve as expert data to learn a control policy. We performed an extensive evaluation of the policy on a physical hybrid arm capable of jointly controlling rigid and soft actuation. We show that with a single policy, the arm is capable of reaching arbitrary poses in the workspace with 3.73 cm ($< 6\%$ overall arm length) and 17.78° error within 12.5 seconds, operating at different control frequencies, and controlling the end effector with different loads. Our results showcase significant improvements in control speed while effectively controlling both the position and orientation of the end effector compared to previous quasi-static controllers for hybrid arms.

2 Introduction

Robotic manipulation requires high dexterity, adaptability, and safe handling of objects to enable intricate manipulations in agroforestry, telehealth, and other applications in constrained settings [2, 3]. Soft continuum arms (SCAs) are inspired by octopus tentacles and elephant trunks and do not have rigid joints and actuators [4–8]. These manipulators are predisposed to work well in such cases due to their adaptability [9–12], but they do not lend to precise operations,

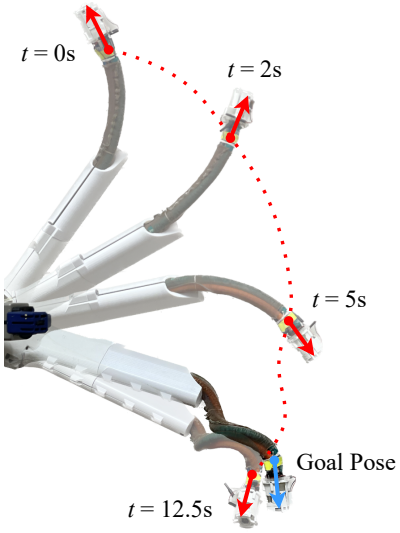

Fig. 1: We developed a non-quasistatic pose controller for a hybrid continuum arm. Our approach enables quick and accurate pose control anywhere in the workspace. The manipulator's inherent dexterity and adaptability make it promising for practical applications [1, 2]. This figure shows a sample evaluation of our controller where it reaches the desired pose within 12.5 seconds at a control frequency of 4 Hz.

*Equal Contribution - Author Order Listed Alphabetically

Journal of Mechanisms and Robotics. Received February 06, 2024;
Accepted manuscript posted February 7, 2025. doi:10.1115/1.4067872

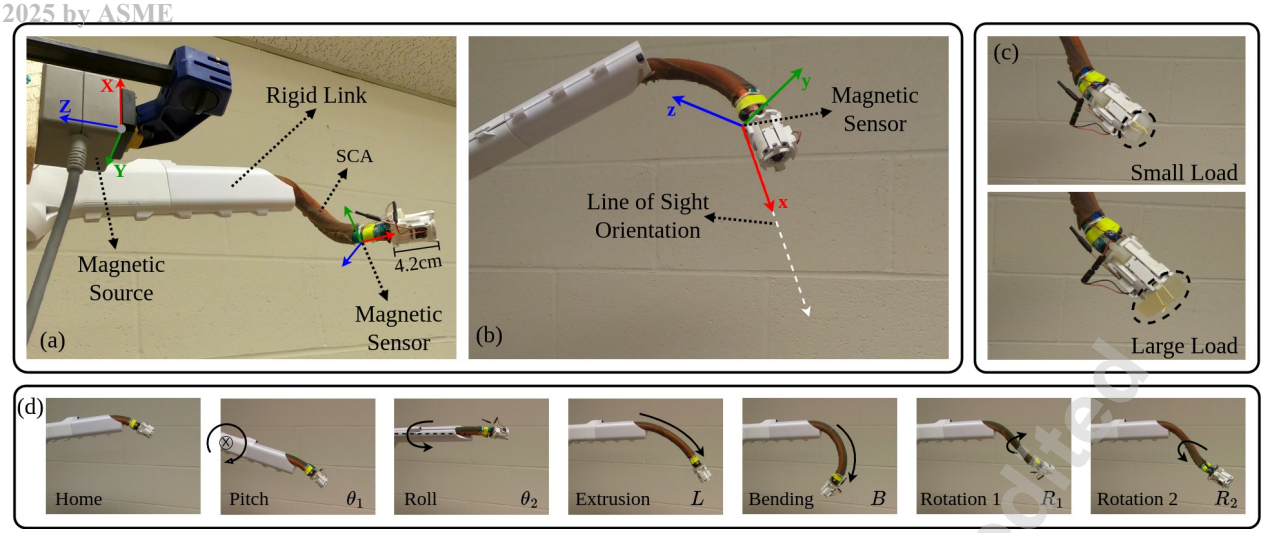

Fig. 2: (a) Overall hybrid manipulator system. The magnetic sensor measures the pose of the end effector relative to the magnetic source. (b) The controller enables control of the end effector's position and line of sight orientation. (c) A small load (3.8 gm, 10% of the SCA's weight) and a large load (10.8 gm, 30% of the SCA's weight) are added at the tip to test the controller's robustness. (d) Each individual actuation of the hybrid arm relative to the home position is shown.

especially with large payloads. Moreover, the workspace and steerability of these manipulators are strongly affected by gravity [13–16]. Also, controlling both end effector position and orientation is difficult given the larger workspace and the under-actuated nature of soft manipulators [17].

On the other hand, hybrid rigid-soft manipulators can be used as a conventional rigid arm when precision and load bearing are deemed important, while a distal deployable soft arm can bring in additional dexterity and adaptability. One such platform is the Variable Length Nested Soft (VaLeNS) [18] configuration, where an SCA is stored in a hollow robotic link with the ability to extrude out of the link as needed. In the VaLeNS configuration, the rigid link positions itself appropriately for the soft arm to extrude and reach the target position with the required end orientation. Its effectiveness has been demonstrated in agricultural berry harvesting [1, 2] and healthcare applications [19] where the soft arm is controlled in a quasi-static manner. While soft arms provide various advantages, a hybrid system offers a broader design space albeit at the cost of increased system complexity. Thus, for practical real-world applications, a fast position and orientation controller must be developed. For instance in the harvesting scenario, without orientation control, reaching the fruit position does not guarantee the gripper is facing the fruit. Additionally, if not done in a timely manner, the economic viability of the system degrades. But due to the rotating base of the hybrid arm, the soft arm is no longer stationary compared to the vertically downward suspended soft arms [20–22]. When coupled with the varying length of the arm and the effects of gravity, more complex shapes and trajectories can be explored that are otherwise unattainable with a purely soft system. Thus, controlling a hybrid rigid-soft manipulator presents more challenges than controlling a purely soft manipulator.

Therefore, we (i) develop a control policy where there is no waiting for the arm to settle unlike in quasistatic controllers (ii) control our hybrid manipulator in the VaLeNS configuration, which is challenging due to its dexterity and increased range of motion (iii) control not only the position but also the orientation of the end effector, which is a more difficult learning problem but has more practical applications in any downstream task that requires grasping (iv) accommodate zero-velocity of the end effector to support downstream manipulation tasks, unlike works like [20] that have dynamic reaching, and (v) extensively evaluate on our physical platform.

3 Related Work

For hybrid arms, a leading approach is to use quasi-static controllers. Dynamic effects are present when operating pneumatic actuators at high speeds. Thus, quasi-static controllers often actuate at low speeds to dissipate these dynamic effects of the system. In this regime, numerical methods like piece-wise constant curvature and Cosserat rod theory can be used to model and control the dynamics of a soft arm [23–25]. [25] uses a Cosserat rod model to control the position and orientation in a quasi-static manner. Using these analytical models or a learned model, [1, 13, 21, 22, 26, 27] learn control policies using a variety of reinforcement learning methods.

The alternative is dynamic controllers, designed to dampen oscillations, considers inertial effects, and other dynamic effects, which can be incorporated either explicitly or learned through dynamic data collection, to improve control at higher frequencies. In many fields, dynamic controllers have high operating frequencies. However, in [28], Jacobian-based dynamic pose control is performed at 1-2 Hz, but their system is actuated using magnetic fields. [29] also uses a magnetically actuated SCA to demonstrate an RNN-based dynamic model to improve upon the real-time prediction of Cosserat rod theory, but this method tends to preserve

the structure of the data without considering the underlying physics leading to overfitting and a lack of generalization. Adding a history of previous states, to better model physical phenomena like hysteresis, could improve these results. Using motion capture techniques for state estimation like many previous works [20, 22, 29–35] is also infeasible for outdoor scenarios due to the unstructured environment and potential occlusions. In contrast, our approach employs a magnetic sensor to track the arm state in real-time offering a more feasible solution for outdoor applications.

[36] uses an adaptive Kalman Filter to enable an optimal controller of a two-section pneumatic arm. However, Kalman filter-based approaches are limited by linear models and Gaussian noise. A popular approach for modeling continuum arms is Cosserat rod theory, but it is computationally expensive, requires parameter estimation, and is restricted to simulation studies [37–39]. Finite-element methods [32] have also been studied for dynamic position control, but struggle with high computational loads slowing real-time deployment. Several Koopman-operator theory-based methods [29–31] seek to overcome this limitation by constructing a linear model in a higher dimensional space, but require large amounts of data leading to wear on the robot, especially when the size of state space increases. Compared with these methods, ours is more sample efficient. [30] uses a similar number of data points as our method to model and control the projected position of a vertically suspended manipulator with 3 actuators. To extend their work, [31] controls a longer arm with variable loading, but the number of samples grows by an order of magnitude. Our method controls the position and orientation of a manipulator of comparable length to that in [31] with 6 actuators with the same magnitude of samples as in [30]. Our system additionally has a mobile soft arm base and a hybrid bending/rotational actuator combination as opposed to purely bending actuators. Thus our method finds a mapping between more complex actuation and task spaces and is more sample efficient. A comparable method on a smaller workspace and actuation space can be seen in [22]. These comparisons are summarized in Table 1. Sample-efficient models combined with reinforcement learning (RL) have been explored [1, 21, 22, 26, 27]. Methods for overcoming the simulation-to-reality gap in RL have also been explored [40–42]. Although these methods are very precise, they require a new policy for each learned trajectory, limiting the scalability of their application. Toward overcoming model degradation issues in order to adapt as a system wears, [43] explores continual learning methods in the soft robot control field under continuously changing loads. However, these approaches involve trade-offs, particularly between avoiding catastrophic forgetting and managing computational and memory overheads.

Many works [31, 44–49] use a model predictive control (MPC) framework for the dynamic control of soft systems but have several limitations. Approaches that model the system as linear [45, 46, 49] offer real-time control but fail to capture the complex non-linearities. While [44] incorporates non-linearities using an RNN-based model followed by an Extended Kalman Filter into the optimization framework but it is limited by the filter's assumptions of noise and modeling. [48] models the dynamics using the Gaussian Process, incorporating uncertainty estimation and propagation, but can be limited computationally as the training data increases. Within the same MPC framework, [47] explores using a genetic evolutionary optimization framework with integral control but can be slow due to the number of iterations to converge particularly as the number of states increases.

In this work, we present a closed-loop controller that controls both the position and line-of-sight orientation of a hybrid rigid-soft continuum manipulator's end effector aimed at real-time deployment in applications where low computational power is available and medium to high operating speeds are necessary. Similar to [20], the forward model is first learned from dynamic data to model our hybrid arm. Zero-velocity tip data is incorporated to enable static reaching and to better support downstream manipulation tasks. Second, the forward model is used to generate optimal trajectories offline to serve as expert data. Third, the generated trajectories are run on the physical robot to avoid the simulation-to-reality gap, and a control policy is learned offline that can be leveraged in a real-time closed-loop controller as shown in Figure 1. This policy is evaluated and analyzed on the physical platform showing a single control policy reaching arbitrary poses in the workspace, being robust to external loads, following trajectories, and operating at different control frequencies. Note that soft arms in previous works are vertically suspended with a fixed orientation of the base and solely have bending actuations [20–22, 26–31, 33, 36, 40]. In contrast, our system varies the orientation of the soft arm base, the soft arm varies in length, and a torsional actuator is considered. As a result, the workspace is not completely symmetric and the effect of gravity is quite severe on the soft portion presenting an additional challenge. Because of the little compute needed at inference time, this policy is ideal in settings like agricultural environments where computing power is limited.

4 Proposed Approach

4.1 Robotic Platform

The robotic platform used is an extrudable BR^2 SCA nested within a 3D-printed rigid arm, also called the VaLeNS arm [18] where the rigid portion can rotate in two degrees of freedom as seen in Figure 2(d). Custom camera and gripper combinations can be mounted on the end of this hybrid arm to support various manipulation tasks. To measure the end effector pose, a Polhemus Patriot Motion Tracker is used. It consists of a pea-sized magnetic sensor mounted at the end effector whose pose is measured relative to a magnetic source placed at the base of the arm as seen in Figure 2(a). The soft arm is made from a combination of one bending and two rotational Fiber Reinforced Elastomeric Enclosures (FREEs) [14, 50], hence the name BR^2 [18]. These are pneumatically pressurized tubes wrapped with an inextensible fiber woven in a helical shape. The angle of the wrapping, the material of the fiber, and the number of fibers affect the behavior of the BR^2 when pressurized [14]. The stepper motor extrudes the soft arm by length, L, using a lead screw

Journal of Mechanisms and Robotics. Received February 06, 2024;
Accepted manuscript posted February 7, 2025. doi:10.1115/1.4067872

Reference	Data Points Sampled	Actuation Space Dimension	Length of Manipulator	Workspace	Soft Manipulator Base is Stationary
[30]	45,103	3 Bending Actuators	34 cm	Controls Laser Points on 2D grid	Yes
[31]	325,733	3 Bending Actuators	70 cm	Controls 3D Position with unknown loads	Yes
[22]	10,000	3 Bending Actuators	44 cm	Control 3D Position	Yes
Ours	41,920	2 Servo Motors, 1 Stepper Motor, 1 Bending Actuator, 2 Rotating Actuators	70 cm	Controls 3D Position and Orientation	No (See θ_1 and θ_2 in Fig. 2)

Table 1: Comparison of sample efficiency with prior methods. [30] has a similar number of data points, but a smaller actuation space, shorter manipulator, smaller workspace, and stationary base making it less sample efficient. [31] uses an order of magnitude more data, but has a smaller actuation space, similar length manipulator, smaller workspace, and a stationary base making it again less sample efficient. No conclusion about sample efficient can be drawn when comparing to [22] because it uses less data, but has a smaller actuation space, shorter manipulator, smaller workspace, and stationary manipulator.

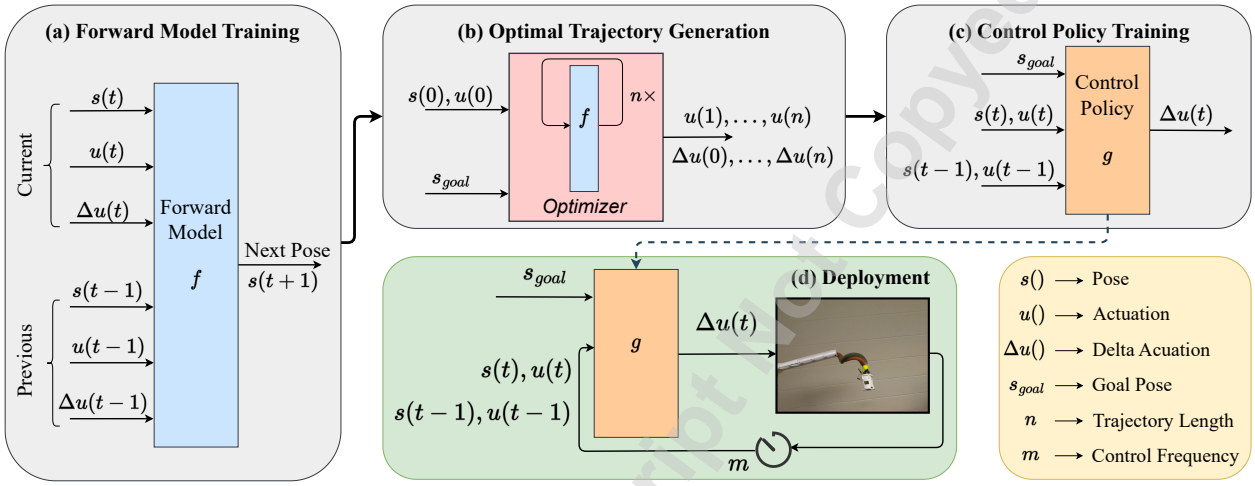

Fig. 3: Pipeline for obtaining and deploying control policy of the hybrid arm. (a) Step one trains the neural-network-based forward model f using data collected on the robot where inputs are poses, actuations, and changes in actuations at current and previous time steps and the output is the pose at the next time step. (b) Step two generates optimal trajectories where the trained forward model f is used iteratively to obtain optimal actuations and optimal changes in actuation in reaching a given initial pose, initial actuation, and goal pose. (c) Step three runs generated trajectories on the robot and trains a control policy g that learns optimal changes in actuation. (d) Finally, the learned offline policy g is deployed on the robot iteratively at m control frequency to reach a goal pose given current and previous poses and actuations.

enclosed within the rigid arm. The rigid arm can adjust its pitch θ_1, and roll θ_2.

Each actuation (Figure 2d) is limited to a set range to support a sufficiently large workspace while ensuring operational safety. The pitch, θ_1, is limited to $[-45°, 45°]$ with 0 being level with the ground. The roll, θ_2, is limited to $[-90°, 90°]$ with 0 being the home position. The extrusion, L, is limited to $[10, 16]$ cm. The bending pressure, B, is limited to $[48.26, 241.32]$ kPa, and the rotating pressures, R_1, R_2, are limited to $[0, 241.32]$ kPa. Both rotational actuators are pressurized to a base pressure of 241.32 kPa and then differentially pressurized to rotate in the desired direction. The differential pressurization serves to minimize oscillations, thereby enhancing the system's load-bearing ability. A single actuation variable can be used to control the pressures in both R_1 and R_2 with differential actuation.

4.2 Control Algorithm

This section elaborates on Figure 3, which describes a pipeline for developing a fast, real-time position and orientation controller. Controlling the orientation is especially important for vision-based tasks where target objects must be in the view of a tip camera as seen in Figure 4. Therefore, the line of sight vector tangent with respect to the arm length, as shown in Figure 2(b), was selected to represent the arm's orientation. This decreases the number of parameters to control and allows freedom of rotation axially along the soft arm as long as the line of sight is maintained. The control policy is developed in three steps offline. First, data is collected on the robot and a forward dynamic model, f, is learned (See Figure 3(a)). Next, optimal trajectories are generated using the forward model (See Figure 3(b)). Finally, a control policy, g, is learned from the generated optimal trajectories (See Figure 3(c)). Here, f and g are neural network-based. We

also verify that the LMU-based model outperforms other architectures (See Tables 2 and 4).

In this paper, the following notations are used. State at time t, $s(t)$, consists of position, $p(t)$, and line of sight orientation, $r(t)$. The line of sight is the x-axis of the sensor's frame (Figure 2b). Actuation is $u = [\theta_1, \theta_2, L, B, R_1, R_2]$ whereas change in actuation is $\Delta u = [\Delta\theta_1, \Delta\theta_2, \Delta L, \Delta B, \Delta R]$. D is the size of the dataset under discussion. The subscript gt will designate the ground truth label.

Forward Model. The forward model data is collected on the robot by first setting the bounds on actuations $(\theta_1, \theta_2, L, B, R_1, R_2)$ as explained in Section 2.1. The bounds on changes in actuations $(\Delta\theta_1, \Delta\theta_2, \Delta L, \Delta B, \Delta R)$ are also set so that the system has sufficient time to respond to the control signals. The changes in the rigid arm joint angles, $\Delta\theta_1$ and $\Delta\theta_2$, were bound between $[-5^\circ, +5^\circ]$ and $[-10^\circ, +10^\circ]$ respectively. The changes in the pressures, ΔB and ΔR, were bound between $[-20.68, 20.68]$ kPa, and changes in the extrusion, ΔL, between $[0, 0.7]$ cm. After random control inputs are actuated for a predetermined time of 5 seconds, the actuations are held constant for 3 seconds to let the system settle as the oscillations dampen and data is collected throughout the 8 seconds at 4 Hz. When operating faster than 4Hz, the set range of actuation changes for the stepper motor would result in high speeds that risk damaging the magnetic sensor cable that goes along the length of the soft arm. This being the most expensive part of the system, a 4Hz operating frequency was selected due to the limitation of the extrusion component. In total, 9600 training samples were collected in this manner, and the training and testing data is split in a 0.8 to 0.2 ratio randomly.

The history of inputs must be considered when modelling the forward model due to hysteresis in the system, a phenomenon in soft robotics that typically occurs in dynamic regimes [17]. A system has hysteresis if the behavior of a system depends on the input and the history of previous states. Given that our approach involves using both current and past states as inputs to the forward model and control policy neural networks, recurrent neural network (RNN)-based architectures [51] are better suited for capturing these temporal dependencies compared to MLPs [52], which lack temporal memory. RNNs [51] are one of the earliest architectures that deal with such time-dependent problems. However, they struggle to capture long-term dependencies and suffer from issues like vanishing and exploding gradients during training. To address these limitations, more advanced architectures such as LSTMs [53] and LMUs [54] have been introduced, which are capable of capturing long-term dependencies. With the data collected, MLP, RNN, LSTM, and LMU-based architectures were trained and tested to evaluate their effectiveness in capturing the temporal relations.

As shown in Figure 3(a), the inputs to the network are the poses, $s(t), s(t-1)$, actuations, $u(t), u(t-1)$, and changes in actuations, $\Delta u(t), \Delta u(t-1)$ at current time step, t, and previous time step, $t-1$, respectively. The output of the forward model is the pose at the next time step, $s(t+1)$. Hyperparameters of these models include memory size, batch size, learning rate, number of layers, hidden layer size, and window length. The overall function of the forward models is represented mathematically in Eq. 1.

$$s(t+1) = f(s(t), u(t), \Delta u(t), \\ s(t-1), u(t-1), \Delta u(t-1)) \tag{1}$$

$$\frac{1}{||D||}\sum_{i=1}^{D} ||\hat{p}^i - p^i_{gt}||^2 - \frac{1}{||D||}\sum_{i=1}^{D} \frac{(\hat{r}^i \cdot r^i_{gt})}{(||\hat{r}^i||||r^i_{gt}||)} \tag{2}$$

The loss function for learning the forward model is composed of the mean square error for learning position $p = (x, y, z)$ and cosine similarity for learning orientation, r which is shown in the Eq. 2. Here, $\hat{p}^i$ and $\hat{r}^i$ denote the (x, y, z) and line of sight orientation prediction from the forward model of ith data in D whereas, p^i_{gt} and r^i_{gt} denote the corresponding ground truth. Ideally, the first term goes to 0 while the second term goes to 1.

Trajectory Optimization. Once the forward model is trained, an optimizer is used to find optimal trajectories for a given start pose, $s(0)$, its corresponding start actuation, $u(0)$, and desired goal pose, s_{goal}, in the workspace as shown in Figure 3(b). These optimal trajectories serve as expert data that the control policy will use to learn the inverse kinematics of the system.

To optimize the trajectory, the Adam optimizer [55] was used. To solve this constrained optimization problem more quickly, a "warm-start" was used where the solution to the unconstrained optimization was used as the initial guess for the constrained optimization [56]. A subset of 10 trajectories were tested to compare optimization with a "cold-start" and "warm-start". Empirically, using a warm start optimization increased speeds by about three fold. With this setting, the Adam optimizer produced more accurate results in fewer iterations. The objective function is shown in Eq. 3 where minimization for a given start and goal state along with the constraints is performed.

α, λ, β, and γ are all manually-tuned hyperparameters that weigh each term in the objective function. The time-horizon, n, is another hyperparameter. The first term in the objective function drives the final position towards the desired goal position. The second term encourages each step of the trajectory to proceed toward the goal position. The third term drives the final orientation to the desired goal orientation. The final term encourages minimal actuation effort across the trajectory. The forward model and the physical limitations of the system act as the constraints. The sequence of actuations, $u(0), \ldots, u(n)$, and the sequence of change in actuations, $\Delta u(0), \ldots, \Delta u(n)$, are the optimization variables. The bounds on the changes in actuation are the same as in the forward model. This prevents the optimizer from exploiting unexplored regions with larger actuations at the cost of limiting the system's ability to dampen oscillatory behavior. The optimizer was run until convergence, and if the solution was not within 1 cm and 25° of the goal, then the optimizer was initialized with a different seed. Optimization would continue as such until convergence of the tolerance or until a

Journal of Mechanisms and Robotics. Received February 06, 2024;
Accepted manuscript posted February 7, 2025. doi:10.1115/1.4067872

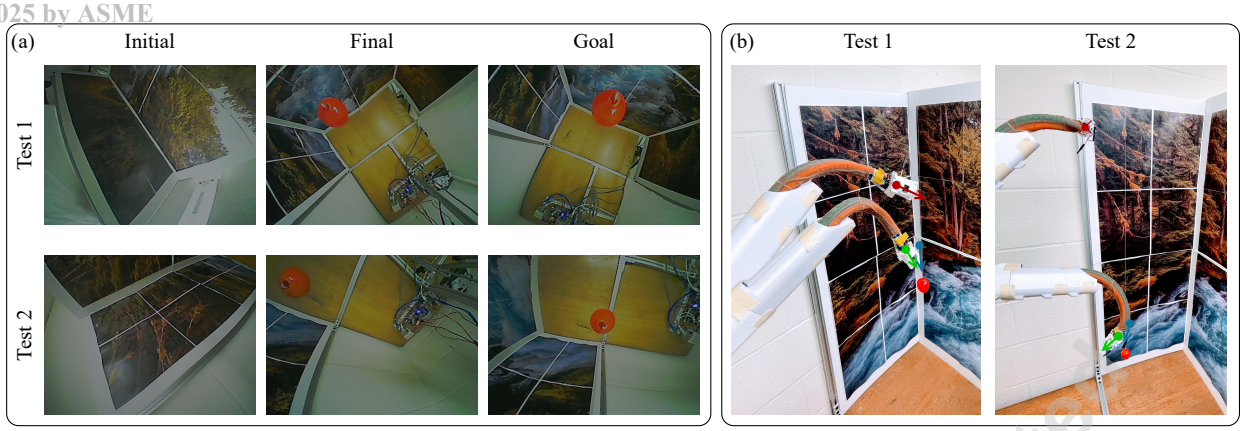

Fig. 4: Bringing target in view of the tip camera using our policy that controls both position and line-of-sight orientation, as seen in (a) tip camera view and (b) third person view, where the start, final, and goal poses are indicated by red, green, and blue arrows respectively.

$$\min_{u(1),\ldots,u(n),\Delta u(0),\ldots,\Delta u(n)} \alpha||p_{goal}-p(n)||^2+\lambda\sum_{t=0}^{n-1}||p_{goal}-p(t)||^2+\beta\Big(1-\frac{r_{goal}\cdot r(n)}{||r_{goal}||||r(n)||}\Big)^2+\gamma\sum_{t=0}^{n}||\Delta u(t)||^2$$
$$\text{where, } u_{min}\le u(t)\le u_{max}, \Delta u_{min}\le \Delta u(t)\le \Delta u_{max}), \forall t\in[0,1,\ldots,n] \quad (3)$$
$$s(t+1)=f(s(t),u(t),\Delta u(t),s(t-1),u(t-1),\Delta u(t-1)), \forall t\in[1,\ldots,n]$$
$$s(t+1)=f(s(t),u(t),\Delta u(t),s(t),u(t),\Delta u(t)), \text{ if } t=0$$

1 hour time limit was reached. To speed up the generation of trajectories, optimization was done in parallel. To summarize, this objective function seeks to minimize the end effector's position error, orientation error, and minimal actuation effort for reaching tasks subject to physical constraints.

Control Policy Network. Solving non-linear optimization may not always lead to convergence, so trajectories with final position errors larger than 10 cm and orientation errors larger than 36° were discarded. About 4% of the 2926 trajectories were discarded and visualized in Figure 5. For clarity, only the desired end pose of the trajectories are plotted. Additionally, the optimizer may have exploited inaccuracies in the learned forward model. So, prior to training the control policy, the generated optimal trajectories are collected on the hybrid arm to eliminate the simulation-to-reality gap, since directly training on simulated optimal trajectories resulted in poor performance 15.8 cm position error and 43.56 deg orientation error when evaluated on 11 test cases of each goal pose each run 5 times.

With the filtered data, various models are trained to learn the inverse kinematics of the system shown in Eq. 4. As seen in Figure 3(c), this policy takes the poses and actuations at current and previous time steps along with goal pose as input. The output is the next optimal change in actuation to progress toward the goal pose. More specifically an RNN, LMU, LSTM, and a 5-layer MLP-based models are trained and compared. To train these policies, the mean square error between predicted change in actuations, $\Delta\hat{u}$, and corresponding ground truth, Δu_{gt}, is used as the loss function as shown in Eq. 5. The optimal trajectories provide Δu_{gt} at each time

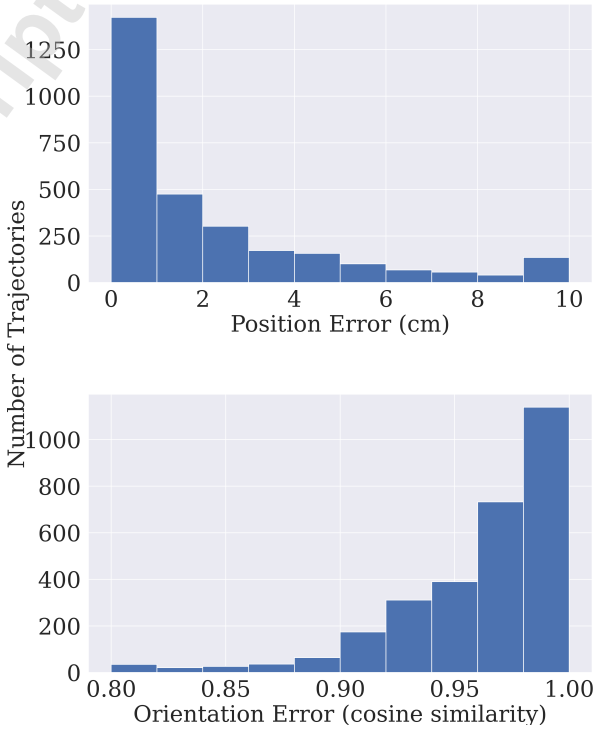

Fig. 5: Visualization of end pose of optimal trajectories kept (blue) and rejected (red) (top). Of all the generated trajectories, 115 trajectories (4%) were filtered due to poor optimization. Note that fewer than 2926 end poses are plotted because multiple trajectories ended at the same final pose but started from a different part of the workspace. A histogram of all their final convergence errors are presented (bottom).

step to reach towards the goal pose, s_{goal}. At inference time, these policies can be used iteratively at an operating control frequency, m Hz, to navigate to a goal position as illustrated in Figure 3(d).

$$\Delta u(t) = g(s_{goal}, s(t), u(t), s(t-1), u(t-1)) \tag{4}$$

$$\frac{1}{||D||}\sum_{i=1}^{D} ||\Delta\hat{u}^i - \Delta u^i_{\text{gt}}||^2 \tag{5}$$

5 Experimental Results

This section describes the suite of tests performed to evaluate the method described in section 4.2. First, the predicted forward model poses were evaluated with the measured pose data. Then, the performance of control policies with different architectures was evaluated on 11 different random poses in the workspace. These tests are repeated with loads attached to the end effector as shown in Figure 2(c) to test the policy's robustness. Additionally, line tracking tests were evaluated with varying control frequency.

5.1 Forward Model Evaluation

The forward models are trained using various architectures and evaluated on 480 unseen test poses as shown in Table 2. The visualization of the training and testing data distribution, which includes the entire trajectory along with the end poses, in the form of relative dexterity measure is shown in Figure 6. To understand the visualization, take for example the xyz-grid areas highlighted in yellow which have a 6/8 dexterity measure in the training data distribution. If we draw a sphere in any of these grid cells, the line-of-sight of our arm-tip can point in 6 directions out of the 8 possible octants in the sphere. The evaluation of forward models on test data in Table 2 indicates the overall learning of dynamics in the workspace with varying dexterity due to the random split of training and testing data as mentioned in Section 4.2. Therefore, the test data which was randomly withheld during training helps evaluate our learned models on unseen data. Note that the sparsity on the right side of Figure 6 results from the 20% random sampling of the entire dataset unseen during training.

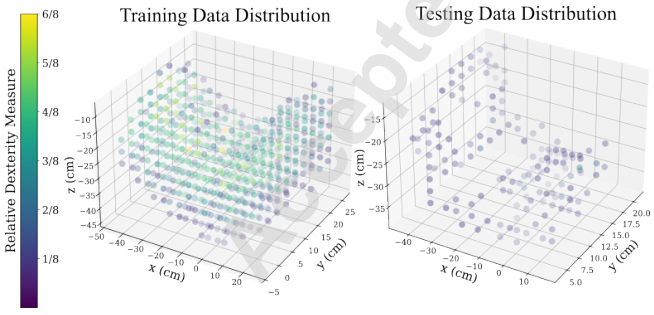

Fig. 6: 3D visualization of training and testing data in the workspace. Each grid cell is colored based on the number of octants covered by the orientation vectors at that cell.

To evaluate the learned forward model, for each trajectory, the mean squared error and and cosine similarity for position and orientation errors are computed over the entire trajectory and averages are calculated. Then, combined average error is computed over the entire test data and reported in the Table 2 for different architectures.

Architecture	Position Error (cm) ↓	Orientation Error (deg) ↓
MLP	1.74	**9.46**
RNN	1.42	10.91
LSTM	1.14	10.13
LMU	**1.13**	10.49

Table 2: Forward model test errors on 480 unseen poses. The LMU has the best position error while the MLP has the best orientation error. See Figure 3(a).

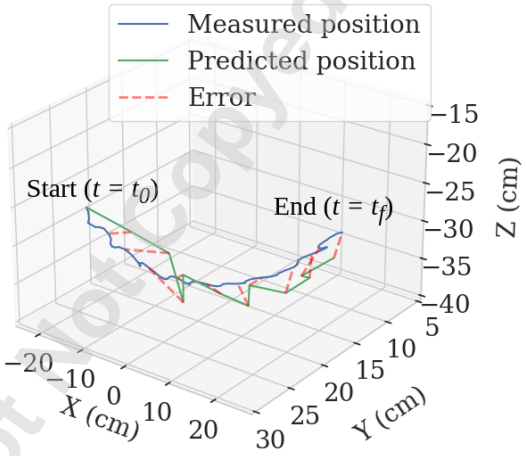

Fig. 7: 3D visualization of the LMU's predicted position of the end effector versus its measured position over time where the forward model predicts the next state based on current and previous states, and control inputs at each time-step.

Each architecture was trained for 200 epochs with a batch size of 100, and a learning rate of $5e^{-4}$ for a fair comparison. The recurrent architectures had a hidden size of 128, and the LMU had a window length of 2. These settings were found using a manual hyperparameter search. Looking at the position and orientation errors together, LSTM and LMU perform better than other models and the LMU model was selected for the trajectory generation as its position error was the lowest. A sample trajectory predicted by forward model based on LMU is shown in Figure 7.

Actuation	Position (cm) ↓	Orientation (deg) ↓
θ_1	0.63 ±0.66	3.74 ±3.94
θ_2	0.74 ±0.78	4.44 ±1.94
L	1.09 ±1.23	5.76 ±4.85
B	1.66 ±1.80	9.87 ±9.03
R	0.42 ±0.45	2.20 ±1.47

Table 3: Variability of pose given an arbitrary actuation. Bending has the highest variance in both position and orientation compared to other actuations.

Journal of Mechanisms and Robotics. Received February 06, 2024;
Accepted manuscript posted February 7, 2025. doi:10.1115/1.4067872

Controller	Avg Final Pose Error		Average Best Position in $x(t)$			Average Best Orientation in $x(t)$		
	Pos (cm) ↓	**Ori** (deg) ↓	**Pos** (cm) ↓	**t** (s) ↓	**Corr Ori** (deg) ↓	**Ori** (deg) ↓	**t** (s) ↓	**Corr Pos** (cm) ↓
Ours (MLP)	7.31	25.65	5.18	8.82	22.44	18.37	8.48	6.93
Ours (RNN)	5.44	21.34	3.85	9.01	22.09	16.5	6.97	7.97
Ours (LSTM)	4.18	21.47	3.01	8.06	23.11	15.47	7.42	**5.53**
Ours (LMU)	**3.73**	17.78	**2.75**	9.15	20.52	**13.13**	8.09	5.92
Ours (LMU) + 3.8gm	4.33	16.85	3.04	8.97	17.7	12.74	8.92	5.44
Ours (LMU) + 10.8gm	5.22	18.34	3.61	8.69	16.76	12.74	6.46	7.05

Table 4: Comparison of policies with different architectures of our learned control policy with respect to position and orientation errors over 11 random start-goal pairs each run 5 times are reported. Additionally, the best position and corresponding time and orientation errors are reported. Similarly, the best orientation error and corresponding time and position error are reported. The LMU-based has the best performance overall. Finally, the LMU control policy performance is evaluated with different loads on the end effector to show robustness to external loads unseen during training.

5.2 System Stochasticity

To baseline the stochasticity of the system, 10 random actuations were sent 5 times each to the system. The range and standard deviation in the measured poses were averaged across the 10 actuations. The results are shown in Table 3, which shows the inherent variability in the hardware.

5.3 Control Policy Evaluation

To construct the control policies, the architectures are trained using 2926 optimal trajectories. Each architecture was trained for 1000 epochs with a batch size of 64 and a learning rate of $1e^{-4}$. The recurrent architectures had a hidden size of 256, and theta was 700 for the LMU. These settings were found manually using a hyperparameter search. A suite of tests are run to evaluate these policies. First, policies are evaluated on 11 different random start-goal pairs each tested 5 times. The LMU is found to have the best performance and is further evaluated by adding external load, tracking trajectories, and determining control frequency robustness.

Reaching Arbitrary Pose. Control policies trained using various architectures are evaluated on 11 random poses. A summary of the results is in Table 4. The control policy was run at 4 Hz with a time horizon of 12.5 seconds. The steady-state position and orientation errors are reported in the first two columns of Table 4. The best position errors, corresponding orientation errors, and their respective times are also provided, as are the best orientation errors, corresponding position errors, and their corresponding times. For each start-goal pair, 5 tests were run to capture the variance. These results baseline the system. In our previous works [26, 57], quasi-static methods obtained results of $1-2$ cm errors in the order of minutes. Our approach sacrifices a small degree of positional accuracy in exchange for significant improvements in control speed and the ability to effectively control end effector orientation. On average, the poses are reached with 3.73 cm and 17.78° error, which is 1.13× the SCA diameter (3.30 cm).

We further evaluate on 2 test cases where the goal pose of a berry-like target is provided such that it is visible from the camera mounted at the tip of the arm. Our control policy is able to make the targets appear in view of the tip camera as seen in Figure 4, using which image-based control methods like visual servoing can be explored in the future to close the gap.

Baseline Comparison. To validate the performance of our proposed approach, we perform several baseline tests. First, we evaluate using random trajectories of the forward model data to directly train the control policy. The results in Table 5 show that the policy trained on the forward model random data has less generalization compared to the policy trained on optimal trajectories. But by leveraging the forward model, generating optimal trajectories helps explore the workspace more efficiently resulting in a better policy.

Second, we have shown that the forward model data collected in a dynamic manner accounts for dynamic effects by running our pipeline shown in Figure 3 but learned on quasistatic data and then testing in a dynamic regime. If no dynamic effects were learned by the policy, then this method would have similar results to a policy trained on dynamic data. However, the results from Table 5 show that the policy trained on data collected in a quasistatic manner does not perform at non-quasistatic operating frequencies. On the other hand, the performance of policy learned on dynamic data does not deviate significantly with increasing operating frequency as shown in the control frequency robustness test in Section 5.3. This shows that our policy learns how to control the system to achieve desired poses quickly although oscillations are not dampened.

Third baseline comparison is done with a model predictive controller (MPC) [58] and a model predictive path integral (MPPI) controller [59] using the learned LMU forward model. Our findings are summarized as follows.

(i) Since MPC operates in real-time, it requires conducting the same nonlinear optimizations as described in Equation 3 at every time step. To enable the 4 Hz operating frequency on a laptop with 16 CPUs running at 2.3GHz and a GeForce RTX 3060 Mobile Graphics card running at 1.4GHz, we chose a time horizon, n, of 2 and a maximum of 20 iterations for optimization. Because of the nonconvexity of the optimization as well as the limited computation time,

Journal of Mechanisms and Robotics. Received February 06, 2024;
Accepted manuscript posted February 7, 2025. doi:10.1115/1.4067872

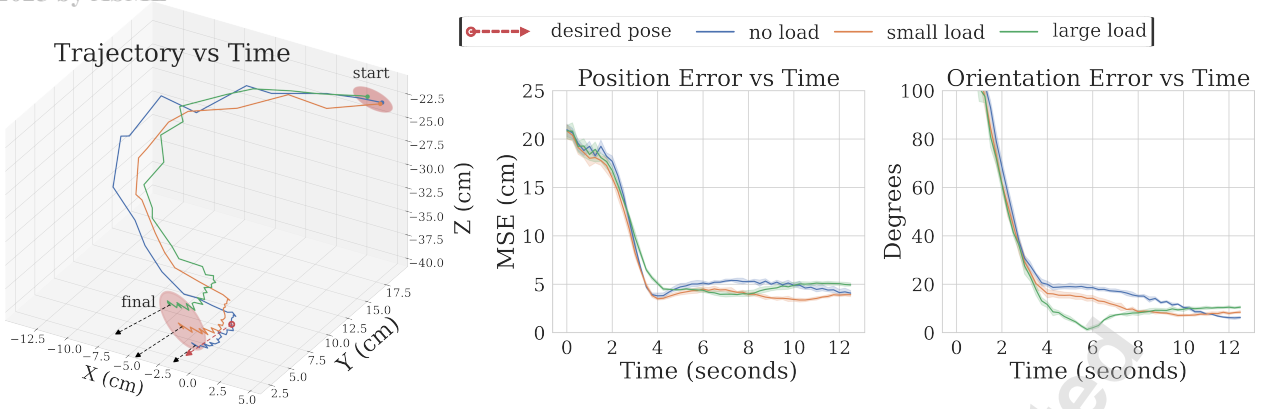

Fig. 8: Comparison of trajectory, position error, and orientation error across different loads. The starting positions change due to different loads, but the position and orientation converge to the desired pose over time.

Controller	Dataset	Control Frequency (Hz)	Average Final Position Error (cm)	Average Final Orientation Error (deg)	Average ITAE Position (min.cm)	Average ITAE Orientation (min.deg)
Learned (LMU)	Random Dynamic	4	18.74	54.28	19.68	58.37
Learned (LMU)	Optimal Quasistatic	0.25	6.48	26.34	7.56	28.00
Learned (LMU)	Optimal Quasistatic	4	9.49	37.13	9.67	43.84
MPC	-	4	27.54	62.35	28.28	62.56
MPPI (1)	-	4	16.28	**17.19**	18.16	**21.48**
MPPI (2)	-	4	8.94	23.12	11.26	23.32
Learned (LMU)	Optimal Dynamic	4	**3.73**	17.78	**6.05**	28.57

Table 5: Baseline comparisons. (Top row) Learned policy trained on random trajectories of forward model data, (rows 2, 3) learned policies trained on optimal trajectories in quasistatic settings tested at quasistatic and dynamic rates, (rows 4, 5, 6) MPC, MPPI (1) and MPPI (2) where optimization is performed online and (bottom row) learned policy on optimal trajectories whose combined position and orientation error is better than the baselines.

the MPC has poor real-time performance as seen in Table 4, though recent studies have explored methods to improve its real-time efficiency [60–62].

(ii) The MPC controller was also implemented in simulation to evaluate its accuracy in a dynamic regime given more compute time. With a time horizon of 20 steps and max iterations of 100k for optimization, a single trajectory required about 3 hours of computation time to reach position accuracy of 1.1 cm and orientation accuracy of 5° with 33 actuations applied in a closed loop fashion. Simulation offers superior accuracy, but computation time and simulation-to-reality errors limit realizing these accuracies on the physical system at the operating frequency of 4 Hz.

(iii) Two MPPI controllers with different weights are also tested as baselines. Each computes 10,000 rollouts each with a time horizon of 15 steps in parallel. MPPI (1) uses the same optimization objective Equation 3, and MPPI (2) weights the position error 4× more. The next actuation is then computed as a weighted average as shown in Equation 6 where S_k is the objective function value of rollout k.

$$\Delta u(t+1) = \frac{\sum_{k=0}^{K} w_k u_k}{\sum_{k=0}^{K} w_k}, w_k = e^{-\frac{S_k}{\lambda}} \tag{6}$$

While their performance is better than MPC, they could not achieve lower position and orientation errors compared to our learned controller because our method learns this difficult nonconvex optimization offline.

Our learned controller outperforms these baselines mainly due to the lack of simulation-to-reality error. The baselines are optimized over a learned forward model, where low data regimes may be exploited that do not translate to the real system. On the contrary, our proposed approach uses op-

Workspace	Average Time to Converge (s)	Convergence Percentage (%)	Average Final Position Error (cm)	Average Final Orientation Error (deg)
$(L,B,R_1,R_2)+(x,y,z)$	4.4	100	2.29 (tol = 2.5)	-
$(L,B,R_1,R_2)+(x,y,z,r_x)$	4.4	90	2.56 (tol=3)	0.86 (tol=0.8)
$(L,B,R_1,R_2)+(x,y,z,r_x)$	7.6	60	2.19 (tol=2.5)	0.90 (tol=0.9)

Table 6: Evaluation of position control and position and orientation control of the soft component of our hybrid system, where their respective policies can take the arm to desired goal pose within $4-7$ seconds.

Control Frequency (Hz)	Best Tolerance with 100% Convergence	Average Trajectory Time (s)	Average Final Position Error (cm)	Average Final Orientation Error (deg)
1	2cm, 5°	15.0 ± 1.1	1.73 ± 0.24	1.89 ± 0.69
2	3cm, 5°	5.5 ± 0.3	2.50 ± 0.53	1.77 ± 0.88
4	3cm, 5°	2.8 ± 0.1	2.58 ± 0.20	1.55 ± 1.29
8	3cm, 5°	1.6 ± 0.0	2.16 ± 0.44	2.49 ± 0.72

Table 7: Control frequency and pose tolerance analysis of line test. The higher the frequency the shorter the trajectory times. Lowering the control frequency results in better performance. Errors and standard deviation are reported for the final position and orientation.

timal trajectories run on the real system to train the learned policy and avoid the simulation-to-reality gap.

Scalability. To evaluate the scalability of learning policies with the number of states, a comparison with two smaller workspaces involving just the soft components (extrusion, bending, and rotation) is performed. The first workspace involves just position control and the second one also involves orientation control, where the pipeline shown in Figure 3 is implemented with appropriate state modifications. Table 6 shows the performance of the two policies. With the full workspace of our system, we require 2926 optimal trajectories to learn a policy that can reach desired poses with an average position error of 3.73 cm and orientation error of 17.78 deg error within 12.5 seconds, compared to 1115 trajectories in both versions smaller workspace in reaching less than 3 cm position and $25-36$ degrees orientation within $4-7$ seconds.

Robustness to External Load. The policy was evaluated with different loading conditions. A small load of 3.8 gm ($\sim$ 10% of the SCA's weight) and a large load of 10.8 gm ($\sim$ 30% of the SCA's weight) were attached to the tip of the arm as seen in Figure 2c. The same 11 trajectories were tested 5 times in these different loading scenarios to capture the variance of the trajectories. As seen in Table 4, the loads have minimal effect on the errors achieved by the policy. Averaging over 55 tests, the final position (column 1) and orientation error (column 2) degrade slightly, but the loads even allow the best positions (column 5) and orientations (column 8) to be achieved more quickly. We attribute this outcome to the damped oscillations due to the larger end effector load. Errors of a test case at different load conditions are shown in Figure 8 showing the robustness of our policy.

Control Frequency Robustness. To further evaluate the policy's robustness, multiple control frequencies were also tested for the line-following trajectory. The line was still segmented into 10 evenly spaced waypoints, but control frequencies of 1, 2, 4, and 8 Hz were evaluated. Frequencies faster than 8 Hz were not attempted due to hardware limitations. The results (See Table 7) show that if a pose can be reached within a certain tolerance, it is reachable with any slower control frequency. For example, if 3 cm and 5° can be converged upon at 8 Hz, then it can also be converged at 1 Hz. This is an interesting finding because the control policy was learned with data collected at 4 Hz.

6 Conclusion

Developing fast, robust, and accurate soft robotic controllers remains an open challenge. In this work, the difficulty of this challenge is increased by tackling orientation as well as position control. Furthermore, the hybrid soft-rigid system considered in this paper is challenging due to variations in the orientation of the soft manipulator base, varying soft manipulator length, and the torsional pneumatic actuator. A dynamic model is first learned and evaluated by using neural networks. Then, optimal trajectories were generated to serve as expert data for training a control policy. This optimal control policy was trained and then evaluated on a physical platform. Its capability of reaching arbitrary poses in the workspace, tracking pose trajectories, and operating at different control frequencies and with unknown loads was analyzed. Arbitrary poses in the workspace were reached within 3.73 cm and 17.78° in 12.5 seconds. The policy was also robust to operating control frequency and minimally affected by external loads without further training or fine-tuning.

While this work demonstrates that accuracy in both position and orientation can be achieved quickly in a complex 3D workspace, there are limitations. Without online learning, our method requires proper data collection to ensure good generalization in all parts of the workspace for both the forward model and the policy. Similarly, if the system changes over time due to issues like wear and tear, online learning must be incorporated to adapt to these changes. Additionally, the policy's performance is limited by the quality of the expert trajectories. Ensuring high-quality non-convex optimizations is difficult with current methods lacking global convergence guarantees. Despite these limitations, our method is a step towards fast and accurate dynamic pose control of soft manipulators. With these promising results, future work will seek to outperform this baseline. Visual feedback can be incorporated if target images or objects are provided. More application-focused extensions like obstacle avoidance and controlling a serial soft manipulator configuration can also be explored.

References

[1] Uppalapati, N. K., Walt, B., Havens, A. J., Mahdian, A., Chowdhary, G., and Krishnan, G., 2020. "A berry picking robot with a hybrid soft-rigid arm: Design and task space control.". In Robotics: Science and Systems.
[2] Chowdhary, G., Gazzola, M., Krishnan, G., Soman, C.,

and Lovell, S., 2019. "Soft robotics as an enabling technology for agroforestry practice and research". *Sustainability,* ***11***(23).

[3] Wen, T., Hu, J., Zhang, J., Li, X., Kang, S., and Zhang, N., 2024. "Design, performance analysis, and experiments of a soft robot for rescue". *Journal of Mechanisms and Robotics,* ***16***(7).

[4] Kadylak, T., Uppalapati, N., Huq, A., Krishnan, G., and Rogers, W. A., 2023. "Engaging healthcare providers to design a robot for telehealth". *Ergonomics in Design*.

[5] Liu, Y., Ge, Z., Yang, S., Walker, I. D., and Ju, Z., 2019. "Elephant's trunk robot: An extremely versatile under-actuated continuum robot driven by a single motor". *Journal of Mechanisms and Robotics,* ***11***(5), p. 051008.

[6] Konda, R., Bombara, D., Chow, E., and Zhang, J., 2024. "Kinematic modeling and open-loop control of a twisted string actuator-driven soft robotic manipulator". *Journal of Mechanisms and Robotics,* ***16***(4).

[7] Banerjee, H., Pusalkar, N., and Ren, H., 2018. "Single-motor controlled tendon-driven peristaltic soft origami robot". *Journal of Mechanisms and Robotics,* ***10***(6), p. 064501.

[8] Xu, Y., Yan, D., Zhang, K., Li, X., Xing, Y., and Shao, L.-H., 2022. "Soft robot based on hyperelastic buckling controlled by discontinuous magnetic field". *Journal of Mechanisms and Robotics,* ***14***(1), p. 011008.

[9] Yip, M. C., and Camarillo, D. B., 2016. "Model-less hybrid position/force control: a minimalist approach for continuum manipulators in unknown, constrained environments". *IEEE Robotics and Automation Letters,* ***1***(2).

[10] Ma, K., Chen, X., Zhang, J., Xie, Z., Wu, J., and Zhang, J., 2023. "Inspired by physical intelligence of an elephant trunk: Biomimetic soft robot with preprogrammable localized stiffness". *IEEE Robotics and Automation Letters,* ***8***(5).

[11] Trivedi, D., Rahn, C. D., Kier, W. M., and Walker, I. D., 2008. "Soft robotics: Biological inspiration, state of the art, and future research". *Applied bionics and biomechanics,* ***5***(3).

[12] Dou, W., Zhong, G., Cao, J., Shi, Z., Peng, B., and Jiang, L., 2021. "Soft robotic manipulators: Designs, actuation, stiffness tuning, and sensing". *Advanced Materials Technologies,* ***6***(9).

[13] Satheeshbabu, S., Uppalapati, N. K., Chowdhary, G., and Krishnan, G., 2019. "Open loop position control of soft continuum arm using deep reinforcement learning". In 2019 International Conference on Robotics and Automation (ICRA), IEEE.

[14] Uppalapati, N. K., and Krishnan, G., 2021. "Design and modeling of soft continuum manipulators using parallel asymmetric combination of fiber-reinforced elastomers". *Journal of Mechanisms and Robotics,* ***13***(1).

[15] George Thuruthel, T., Falotico, E., Manti, M., Pratesi, A., Cianchetti, M., and Laschi, C., 2017. "Learning closed loop kinematic controllers for continuum manipulators in unstructured environments". *Soft robotics,* ***4***(3).

[16] Rolf, M., and Steil, J. J., 2013. "Efficient exploratory learning of inverse kinematics on a bionic elephant trunk". *IEEE transactions on neural networks and learning systems,* ***25***(6).

[17] George Thuruthel, T., Ansari, Y., Falotico, E., and Laschi, C., 2018. "Control strategies for soft robotic manipulators: A survey". *Soft robotics,* ***5***(2).

[18] Uppalapati, N. K., and Krishnan, G., 2020. "Valens: Design of a novel variable length nested soft arm". *IEEE Robotics and Automation Letters,* ***5***(2).

[19] Uppalapati, N. K., Kadylak, T., Rogers, W., and Krishnan, G. "Morphological switching robots to support independent living for older adults".

[20] Thuruthel, T. G., Falotico, E., Renda, F., and Laschi, C., 2018. "Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators". *IEEE Transactions on Robotics,* ***35***(1).

[21] Gan, Y., Li, P., Jiang, H., Wang, G., Jin, Y., Chen, X., and Ji, J., 2022. "A reinforcement learning method for motion control with constraints on an hpn arm". *IEEE Robotics and Automation Letters,* ***7***(4).

[22] Centurelli, A., Arleo, L., Rizzo, A., Tolu, S., Laschi, C., and Falotico, E., 2022. "Closed-loop dynamic control of a soft manipulator using deep reinforcement learning". *IEEE Robotics and Automation Letters,* ***7***(2).

[23] Escande, C., Chettibi, T., Merzouki, R., Coelen, V., and Pathak, P. M., 2014. "Kinematic calibration of a multi-section bionic manipulator". *IEEE/ASME transactions on mechatronics,* ***20***(2).

[24] Webster III, R. J., and Jones, B. A., 2010. "Design and kinematic modeling of constant curvature continuum robots: A review". *The International Journal of Robotics Research,* ***29***(13).

[25] Shi, J., Abad Guaman, S., Dai, J., and Wurdemann, H., 2023. "Position and orientation control for hyperelastic multi-segment continuum robots". *IEEE/ASME Transactions on Mechatronics*.

[26] Satheeshbabu, S., Uppalapati, N. K., Fu, T., and Krishnan, G., 2020. "Continuous control of a soft continuum arm using deep reinforcement learning". In 2020 3rd IEEE International Conference on Soft Robotics (RoboSoft), IEEE.

[27] Wu, Q., Gu, Y., Li, Y., Zhang, B., Chepinskiy, S. A., Wang, J., Zhilenkov, A. A., Krasnov, A. Y., and Chernyi, S., 2020. "Position control of cable-driven robotic soft arm based on deep reinforcement learning". *Information,* ***11***(6).

[28] Lin, D., Chen, W., He, K., Jiao, N., Wang, Z., and Liu, L., 2022. "Position and orientation control of multisection magnetic soft microcatheters". *IEEE/ASME Transactions on Mechatronics,* ***28***(2).

[29] Haggerty, D. A., Banks, M. J., Kamenar, E., Cao, A. B., Curtis, P. C., Mezić, I., and Hawkes, E. W., 2023. "Control of soft robots with inertial dynamics". *Science Robotics,* ***8***(81).

[30] Bruder, D., Fu, X., Gillespie, R. B., Remy, C. D.,

and Vasudevan, R., 2020. "Data-driven control of soft robots using koopman operator theory". *IEEE Transactions on Robotics,* ***37***(3).

[31] Bruder, D., Fu, X., Gillespie, R. B., Remy, C. D., and Vasudevan, R., 2021. "Koopman-based control of a soft continuum manipulator under variable loading conditions". *IEEE robotics and automation letters,* ***6***(4).

[32] Katzschmann, R. K., Thieffry, M., Goury, O., Kruszewski, A., Guerra, T.-M., Duriez, C., and Rus, D., 2019. "Dynamically closed-loop controlled soft robotic arm using a reduced order finite element model with state observer". In 2019 2nd IEEE international conference on soft robotics (RoboSoft), IEEE.

[33] Fischer, O., Toshimitsu, Y., Kazemipour, A., and Katzschmann, R. K., 2023. "Dynamic task space control enables soft manipulators to perform real-world tasks". *Advanced Intelligent Systems,* ***5***(1).

[34] Gillespie, M. T., Best, C. M., Townsend, E. C., Wingate, D., and Killpack, M. D., 2018. "Learning nonlinear dynamic models of soft robots for model predictive control with neural networks". In 2018 IEEE International Conference on Soft Robotics (RoboSoft), IEEE.

[35] Tariverdi, A., Venkiteswaran, V. K., Richter, M., Elle, O. J., Tørresen, J., Mathiassen, K., Misra, S., and Martinsen, Ø. G., 2021. "A recurrent neural-network-based real-time dynamic model for soft continuum manipulators". *Frontiers in Robotics and AI,* ***8***, p. 631303.

[36] Li, M., Kang, R., Branson, D. T., and Dai, J. S., 2017. "Model-free control for continuum robots based on an adaptive kalman filter". *IEEE/ASME Transactions on Mechatronics,* ***23***(1).

[37] Doroudchi, A., and Berman, S., 2021. "Configuration tracking for soft continuum robotic arms using inverse dynamic control of a cosserat rod model". In 2021 IEEE 4th International Conference on Soft Robotics (RoboSoft), IEEE.

[38] Thuruthel, T. G., Falotico, E., Renda, F., and Laschi, C., 2017. "Learning dynamic models for open loop predictive control of soft robotic manipulators". *Bioinspiration & biomimetics,* ***12***(6).

[39] Alqumsan, A. A., Khoo, S., and Norton, M., 2019. "Robust control of continuum robots using cosserat rod theory". *Mechanism and Machine Theory,* ***131***.

[40] Jitosho, R., Lum, T. G. W., Okamura, A., and Liu, K., 2023. "Reinforcement learning enables real-time planning and control of agile maneuvers for soft robot arms". In Conference on Robot Learning, PMLR.

[41] Nazeer, M. S., Bianchi, D., Campinoti, G., Laschi, C., and Falotico, E., 2023. "Policy adaptation using an online regressing network in a soft robotic arm". In 2023 IEEE International Conference on Soft Robotics (RoboSoft), IEEE, pp. 1–7.

[42] Nazeer, M. S., Laschi, C., and Falotico, E., 2024. "Rl-based adaptive controller for high precision reaching in a soft robot arm". *IEEE Transactions on Robotics*.

[43] Piqué, F., Kalidindi, H. T., Fruzzetti, L., Laschi, C., Menciassi, A., and Falotico, E., 2022. "Controlling soft robotic arms using continual learning". *IEEE Robotics and Automation Letters,* ***7***(2), pp. 5469–5476.

[44] Preiss, J. A., Millard, D., Yao, T., and Sukhatme, G. S., 2022. "Tracking fast trajectories with a deformable object using a learned model". In 2022 International Conference on Robotics and Automation (ICRA), IEEE, pp. 1351–1357.

[45] Huang, Y., Hofer, M., and D'Andrea, R., 2021. "Offset-free model predictive control: A ball catching application with a spherical soft robotic arm". In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp. 563–570.

[46] Céspedes, A., Terreros, R., Morales, S., Huamaní, A., Fabián, J., and Canahuire, R., 2024. "Model predictive control of a soft laparoscope using neural networks". In 2024 2nd International Conference on Mechatronics, Control and Robotics (ICMCR), IEEE, pp. 57–61.

[47] Jensen, S., Salmon, J. L., and Killpack, M. D., 2024. "Model evolutionary gain-based predictive control (mega-pc) for soft robotics". In 2024 IEEE 7th International Conference on Soft Robotics (RoboSoft), IEEE, pp. 816–823.

[48] Tang, Z., Xin, W., Wang, P., and Laschi, C., 2024. "Learning-based control for soft robot–environment interaction with force/position tracking capability". *Soft Robotics*.

[49] Hyatt, P., Wingate, D., and Killpack, M. D., 2019. "Model-based control of soft actuators using learned non-linear discrete-time models". *Frontiers in Robotics and AI,* ***6***, p. 22.

[50] Singh, G., and Krishnan, G., 2017. "A constrained maximization formulation to analyze deformation of fiber reinforced elastomeric actuators". *Smart Materials and Structures,* ***26***(6).

[51] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 1985. Learning internal representations by error propagation. Tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science.

[52] Haykin, S., 1998. *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

[53] Hochreiter, S., and Schmidhuber, J., 1997. "Long short-term memory". *Neural computation,* ***9***(8).

[54] Voelker, A., Kajić, I., and Eliasmith, C., 2019. "Legendre memory units: Continuous-time representation in recurrent neural networks". *Advances in neural information processing systems,* ***32***.

[55] Kingma, D. P., and Ba, J., 2014. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980*.

[56] Sambharya, R., Hall, G., Amos, B., and Stellato, B., 2024. "Learning to warm-start fixed-point optimization algorithms". *Journal of Machine Learning Research,* ***25***(166), pp. 1–46.

[57] Kamtikar, S., Marri, S., Walt, B., Uppalapati, N. K., Krishnan, G., and Chowdhary, G., 2022. "Visual servoing for pose control of soft continuum arm in a structured environment". *IEEE Robotics and Automation Letters,* ***7***(2).

[58] Garcia, C. E., Prett, D. M., and Morari, M., 1989. "Model predictive control: Theory and practice—a survey". *Automatica,* ***25***(3).

[59] Williams, G., Aldrich, A., and Theodorou, E. A., 2017. "Model predictive path integral control: From theory to parallel computation". *Journal of Guidance, Control, and Dynamics,* ***40***(2).

[60] Alonso, C. A., and Tseng, S.-H., 2022. "Effective gpu parallelization of distributed and localized model predictive control". In 2022 IEEE 17th International Conference on Control & Automation (ICCA), IEEE, pp. 199–206.

[61] Adabag, E., Atal, M., Gerard, W., and Plancher, B., 2024. "Mpcgpu: Real-time nonlinear model predictive control through preconditioned conjugate gradient on the gpu". In 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 9787–9794.

[62] Lee, Y., Choi, K. H., and Kim, K.-S., 2024. "Gpu-enabled parallel trajectory optimization framework for safe motion planning of autonomous vehicles". *IEEE Robotics and Automation Letters*.

7 Figure and Table Captions

Figure 1 Caption: We developed a non-quasistatic pose controller for a hybrid continuum arm. Our approach enables quick and accurate pose control anywhere in the workspace. The manipulator's inherent dexterity and adaptability make it promising for practical applications [1,2]. This figure shows a sample evaluation of our controller where it reaches the desired pose within 12.5 seconds at a control frequency of 4 Hz.

Figure 2 Caption: (a) Overall hybrid manipulator system. The magnetic sensor measures the pose of the end effector relative to the magnetic source. (b) The controller enables control of the end effector's position and line of sight orientation. (c) A small load (3.8 gm, 10% of the SCA's weight) and a large load (10.8 gm, 30% of the SCA's weight) are added at the tip to test the controller's robustness. (d) Each individual actuation of the hybrid arm relative to the home position is shown.

Figure 3 Caption: Pipeline for obtaining and deploying control policy of the hybrid arm. (a) Step one trains the neural-network-based forward model f using data collected on the robot where inputs are poses, actuations, and changes in actuations at current and previous time steps and the output is the pose at the next time step. (b) Step two generates optimal trajectories where the trained forward model f is used iteratively to obtain optimal actuations and optimal changes in actuation in reaching a given initial pose, initial actuation, and goal pose. (c) Step three runs generated trajectories on the robot and trains a control policy g that learns optimal changes in actuation. (d) Finally, the learned offline policy g is deployed on the robot iteratively at m control frequency to reach a goal pose given current and previous poses and actuations.

Figure 4 Caption: Bringing target in view of the tip camera using our policy that controls both position and line-of-sight orientation, as seen in (a) tip camera view and (b) third person view, where the start, final, and goal poses are indicated by red, green, and blue arrows respectively.

Figure 5 Caption: Visualization of end pose of optimal trajectories kept (blue) and rejected (red) (top). Of all the generated trajectories, 115 trajectories (4%) were filtered due to poor optimization. Note that fewer than 2926 end poses are plotted because multiple trajectories ended at the same final pose but started from a different part of the workspace. A histogram of all their final convergence errors are presented (bottom).

Figure 6 Caption: 3D visualization of training and testing data in the workspace. Each grid cell is colored based on the number of octants covered by the orientation vectors at that cell.

Figure 7 Caption: 3D visualization of the LMU's predicted position of the end effector versus its measured position over time where the forward model predicts the next state based on current and previous states, and control inputs at each time-step.

Figure 8 Caption: Comparison of trajectory, position error, and orientation error across different loads. The starting positions change due to different loads, but the position and orientation converge to the desired pose over time.

Table 1 Caption: Comparison of sample efficiency with prior methods. [30] has a similar number of data points, but a smaller actuation space, shorter manipulator, smaller workspace, and stationary base making it less sample efficient. [31] uses an order of magnitude more data, but has a smaller actuation space, similar length manipulator, smaller workspace, and a stationary base making it again less sample efficient. No conclusion about sample efficient can be drawn when comparing to [22] because it uses less data, but has a smaller actuation space, shorter manipulator, smaller workspace, and stationary manipulator.

Table 2 Caption: Forward model test errors on 480 unseen poses. The LMU has the best position error while the MLP has the best orientation error. See Figure 3(a).

Table 3 Caption: Variability of pose given an arbitrary actuation. Bending has the highest variance in both position and orientation compared to other actuations.

Table 4 Caption: Comparison of policies with different architectures of our learned control policy with respect to position and orientation errors over 11 random start-goal pairs each run 5 times are reported. Additionally, the best position and corresponding time and orientation errors are reported. Similarly, the best orientation error and corresponding time and position error are reported. The LMU-based has the best performance overall. Finally, the LMU control policy performance is evaluated with different loads on the end effector to show robustness to external loads unseen during training.

Table 5 Caption: Baseline comparisons. (Top row) Learned policy trained on random trajectories of forward model data, (rows 2, 3) learned policies trained on optimal trajectories in quasistatic settings tested at quasistatic and dynamic rates, (rows 4, 5, 6) MPC, MPPI (1) and MPPI (2) where optimization is performed online and (bottom row) learned policy on optimal trajectories whose combined position and orientation error is better than the baselines.

Table 6 Caption: Evaluation of position control and position and orientation control of the soft component of our hybrid system, where their respective policies can take the arm to desired goal pose within $4-7$ seconds.

Table 7 Caption: Control frequency and pose tolerance analysis of line test. The higher the frequency the shorter the trajectory times. Lowering the control frequency results in better performance. Errors and standard deviation are reported for the final position and orientation.