Tracking Fast Trajectories with a Deformable Object using a Learned Model

James A. Preiss*, David Millard*, Tao Yao*, Gaurav S. Sukhatme†

***Abstract*— We propose a method for robotic control of deformable objects using a learned nonlinear dynamics model. After collecting a dataset of trajectories from the real system, we train a recurrent neural network (RNN) to approximate its input-output behavior with a latent state-space model. The RNN internal state is low-dimensional enough to enable realtime nonlinear control methods. We demonstrate a closed-loop control scheme with the RNN model using a standard nonlinear state observer and model-predictive controller. We apply our method to track a highly dynamic trajectory with a point on the deformable object, in real time and on real hardware. Our experiments show that the RNN model captures the true system's frequency response and can be used to track trajectories outside the training distribution. In an ablation study, we find that the full method improves tracking accuracy compared to an open-loop version without the state observer.**

I. Introduction and Related Work

Manipulating deformable objects represents a challenging area of robotics. In contrast to typical objects consisting of a single rigid body, deformable objects often admit limited control authority and have dynamics that are difficult to predict. At the same time, many objects of human interest are deformable. Safe, reliable robotic manipulation of these objects is critical for capable general-purpose robots [1], [2].

As with many manipulation problems in robotics, there are multiple ways to model the dynamical behavior of the object in question. Broadly speaking, there are two classes of models: Fully data-driven methods based on an expressive function class with many parameters, and analytical methods based on a physical model with fewer parameters that can be identified from data.

Physics-based models of deformable objects have been studied extensively in science and engineering contexts, and many constitutive models have been described for various materials [3]. These models are continuous in space and time, and simulation on computer hardware requires a discretization strategy. Finite element methods (FEM) have been used for decades to solve continuum mechanics problems [4] and have shown promise for robotic control. Recent work with FEM modeling addresses dynamic control of deformable objects and soft robots with offline trajectory optimization [5]–[10]. FEM is appealing as it admits extensive theoretical analysis [11], [12]. For real-world problems, it is possible to estimate the parameters of FEM meshes in a principled way from exteroceptive sensors common in robotics [13]. Alternative discretizations include the material point method [14], [15], (extended) position based dynamics [16], and meshless shape matching [17]. Additionally, task-specific reduced states can be formulated, which accelerate control, perception and planning [18], [19].

Deformable objects can be complex to model, and for objects with resonant dynamics, seemingly minor errors in model assumptions or material parameter estimates can cause large deviations in dynamic behavior over time [20]. For these reasons, purely data-driven methods are an appealing alternative to physics-based models. Machine learning methods have been explored in deformable manipulation [21], for purely kinematic trajectory tracking [20], for cable-driven soft robot actuators [22], for control of pneumatic deformable mechanisms [23], and for structures actuated by shape-memory alloys [24]. For scenarios where controllers can continuously interact with their environment to improve, model-based reinforcement learning has been proposed [25]. Other data-driven approaches studied in soft robot control include proper orthogonal decompositions [26].

In this work, we propose a data-driven method for modeling and trajectory-tracking control of a deformable object. Tracking fast trajectories serves as a benchmark for a method's ability to predict and account for dynamics in the manipulated object. This is of particular interest for free-flying space robots like the Astrobee platform [27], where the passive dynamics of many objects of interest are much stronger than the actuator's control authority. Our method models the dynamics in with a long short-term memory (LSTM) recurrent neural network (RNN) [28], [29], trained in a standard sequence modeling setup using input-output trajectory data from a real physical system. The internal state of the learned RNN is not physically meaningful, but the RNN still forms a discrete-time dynamical system that is compatible with standard methods in state-space nonlinear control. We therefore apply model-predictive control [30] and extended Kalman filtering methods [31] to track the trajectory using the RNN model.

We implement our method to run online with a real robot, and use it to "draw in the air" along a fast trajectory with the free end of a pool noodle (long foam cylinder) held by the robot arm. To measure how well non-instantaneous dynamics are captured, we compare frequency response plots of the true system and the RNN model. To verify that a nonlinear observer helps compensate for model errors, we perform an ablation study comparing our method to an open-loop variant where the RNN model is assumed to be perfect. We find that the full closed-loop method reduces trajectory tracking error.

All authors are affiliated with the University of Southern California.

* Equal contribution. {japreiss,dmillard,taoyao}@usc.edu

† G.S. Sukhatme holds concurrent appointments as a Professor at USC and as an Amazon Scholar. This paper describes work performed at USC and is not associated with Amazon.

This work was supported in part by a NASA Space Technology Research Fellowship, grant number 80NSSC19K1182.

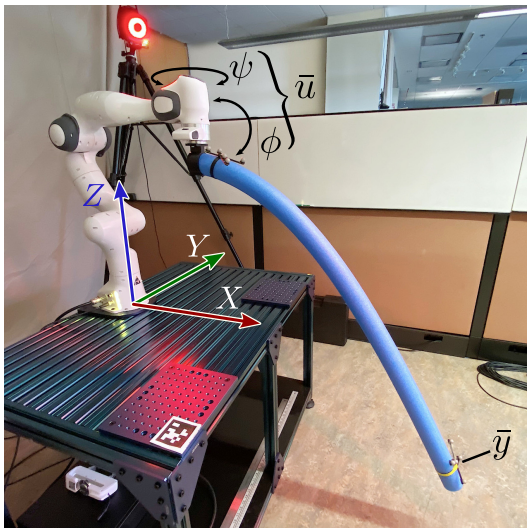

Fig. 1. Physical testbed for our method. Trajectories are tracked by a Vicon motion capture marker attached to the end of a pool noodle, rigidly held by a Franka Emika Panda robot. Pitch/yaw inputs $\bar{u} = (\phi, \psi)$, tracking point measurement $\bar{y}$, and coordinate axes (X, Y, Z) shown. (The fiducial marker visible in the image is not used.)

RNNs have a long history within the broader field of nonlinear state-space system identification with partial observability [32]. Such models can be used as an intermediate tool for learning a control policy, e.g. in model-based reinforcement learning [33], or directly with nonlinear observers and controllers as in our method [34]. To our knowledge, our work represents the first application of the RNN/observer/MPC architecture to the task of manipulating deformable objects.

II. Problem Setting and Preliminaries

We consider a robot manipulating a deformable object such that a particular point on the object tracks a trajectory. We use the following notation: The special Euclidean group of rigid transformations in three-dimensional space is denoted by $SE(3)$. The notation $A \succeq 0$ (resp. $A \succ 0$) indicates that the matrix A is positive semidefinite (resp. definite). The notation $\|y\|_W = \sqrt{y^\top W y}$ for $W \succeq 0$ indicates a weighted Euclidean norm. The notation $\mathrm{diag}(x)$ refers to a diagonal matrix with the entries of the vector x on its diagonal. We define a setting with the following assumptions:

Assumption 2.1 (Input): The deformable object is attached to the robot's end effector by an unbreakable grasp at a fixed position. The control policy interacts with the robot by commanding the position and attitude of the grasp point.

Assumption 2.2 (Output): The robot's perception system can accurately measure the three-dimensional position of a single point on the surface of the deformable object.

Assumption 2.3 (Objective): The robot should manipulate the deformable object such that the measured point on its surface tracks a given trajectory in three-dimensional space.

Assumption 2.4 (Protocol): Interaction with the environment is divided into two stages. In the *preparation* stage, the robot can interact with the deformable object, perform calculations, and store results. There is no time limit on this stage. In the *testing* stage, a supervisor reveals the trajectory to be tracked. The robot must track the trajectory promptly without slow pre-computations. The robot can use all the results stored from the preparation stage, but cannot interact with the deformable object in any way other than attempting to track the trajectory. This restriction is motivated by safety- and time-constrained tasks, where it may not be feasible for the robot to perform additional exploratory actions.

Formally, we assume that the coupled system of the robot and deformable object can be described by a continuous-time, deterministic, time-invariant dynamical system

$$\dot{\bar{x}} = \bar{f}(\bar{x}, \bar{u}), \quad \bar{y} = \bar{h}(\bar{x}), \tag{1}$$

where the state $\bar{x}$ is an abstract infinite-dimensional quantity representing the full continuum state of the deformable object and the robot, the input $\bar{u} \in SE(3)$ is the desired pose of the robot end effector, and the output $\bar{y} \in \mathbb{R}^3$ is the position of the measured point. The unknown dynamics $\bar{f}$ encapsulates both the deformable object itself and a low-level controller that attempts to track the end effector pose input $\bar{u}$ by issuing actuator commands, e.g. motor torques, to the robot. The measurement model $\bar{h}$ extracts the position of the measured point from the continuum state.

Assumption 2.5: The true system has a unique globally exponentially stable equilibrium state $\bar{x}_0$ corresponding to the steady-state identity input I, i.e. $\bar{f}(\bar{x}_0, I) = 0$. This assumption is realistic for damped elastic objects, such as closed cell foam. We assume that no new plastic deformations to the foam (such as a permanent bend) are introduced during our experimental protocol.

The control policy interacts with the system (1) in discrete time steps of fixed length Δ_t. In this paragraph we overload the same notations for discrete-time and continuous-time inputs, states, and outputs; for the remainder of the paper we refer to the discrete-time quantities exclusively. At step $k \in \mathbb{N}$, the discrete-time input $\bar{u}[k]$ is supplied to the continuous-time system as a zero-order hold, resulting in the continuous-time input signal $\bar{u}(t) = \bar{u}[\lfloor t/\Delta_t \rfloor]$ for all $t \geq 0$, and the discrete-time output $\bar{y}[k]$ is sampled such that $\bar{y}[k] = \bar{y}(k\Delta_t)$. Our control task is specified by a discrete-time signal of K goal positions $z[1], \ldots, z[K] \in \mathbb{R}^3$ for the tracked point and the cost function

$$J = \textstyle\sum_{k=1}^{K} \|z[k] - \bar{y}[k]\|_W^2, \tag{2}$$

where the weighting matrix $W \succeq 0$ encodes a (potentially) non-isotropic tracking objective.

Remark: The protocol in Assumption 2.4 rules out some methods that have been widely used for deformable manipulation. Trajectory optimization methods that build a local dynamics model around a reference trajectory by interacting with the environment, such as guided policy search [25], violate the rule against non-task interaction in the testing stage.

III. Methods

The components of our method are outlined in Figure 2. To ensure that our system relies on the whip-like resonant

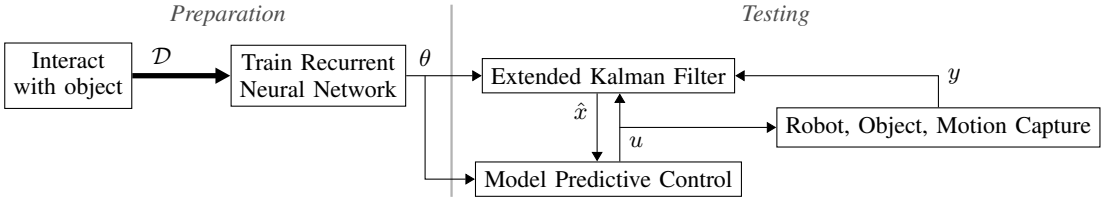

Fig. 2. Diagram of our system. A small dataset of control inputs and tracked marker locations is obtained from the real system. A recurrent neural network (RNN) model is trained to predict input/output behavior with a latent state. The RNN forms the nonlinear dynamics model for an extended Kalman filter (EKF) and model-predictive controller (MPC) to track a dynamic input trajectory with the deformable object.

behavior of the deformable object to track trajectories, rather than relying on large translational movements of the robot end effector, we restrict the inputs to only the pitch ϕ and yaw ψ angles of the end effector. We denote this restricted space as $\mathcal{U} \subset SE(3)$. In practice, we generate training data within a compact subset of pitch and yaw angles, and regularize the MPC problem so that inputs far from the identity I are penalized. Therefore, we parameterize $\mathcal{U}$ by $\mathbb{R}^2$ (in radians) and ignore potential issues of multiple covering.

A. Data collection

We begin by collecting a training dataset $\mathcal{D}$ containing N input-output trajectories from the real system. We denote by $\bar{u}[j,k]$ and $\bar{y}[j,k]$ the input and output from the k^{th} time step of the j^{th} trajectory in $\mathcal{D}$. Before starting each trajectory, we apply the identity input and allow the system to settle for several seconds (10 in our experiments) to ensure that the system returns to a state very close to the rest state $\bar{x}_0$, which will happen promptly due to Assumption 2.5. We then apply random sinusoidal pitch and yaw inputs. Sinusoidal inputs excite the system enough to demonstrate large-scale dynamics such as resonance that are important for manipulation, but are smooth enough to respect the actuator limits of our robot.

The sinusoidal inputs are chosen to respect a user-selected angular acceleration limit $\dot{\omega}_{\max}$ and absolute angle limits $\phi_{\min}, \phi_{\max}$ and $\psi_{\min}, \psi_{\max}$ to avoid triggering our robot arm's built-in safety stop when actuator limits are reached. For each angle, the sinusoid's frequency ν is sampled log-uniformly between $1/8\,\text{Hz}$ and $3\,\text{Hz}$. The maximum angular acceleration of a sinusoid is given by $2\pi A\nu^2$, where A is the amplitude. Therefore, ν and the acceleration limit induce a maximum amplitude $A \leq A_{\max} = \dot{\omega}_{\max}/2\pi\nu^2$. We sample the amplitude uniformly from $[0.1, A_{\max}]$. Finally, we sample the phase uniformly from $[0, 2\pi)$ and add a constant offset, which is sampled uniformly from the range induced by the amplitude and angle limits.

B. RNN dynamics model

The true state $\bar{x}$ of the deformable object is an infinite-dimensional continuum of points, which is not representable on a computer without discretization and approximation. Furthermore, the dynamics of the system are not purely dictated by the behavior of the deformable object—they also include the behavior of the low-level controller of the robot arm. We overcome both challenges by representing the system state as the hidden state of a learned RNN. In comparison to other methods of system identification from input/output data, such as linear methods [35], RNNs are a highly expressive class of nonlinear state-space models. The RNN is a generic function approximation scheme parameterized by a real-valued vector θ, and consists of a discrete-time dynamics model

$$x[k+1] = f_\theta(x[k], u[k]), \quad y[k] = h_\theta(x[k]), \tag{3}$$

where $x \in \mathbb{R}^n$ is the internal state, u is the input, y is the output, and f_θ and h_θ are the dynamics and measurement functions respectively. Both f_θ and h_θ are differentiable with respect to their arguments and the parameter θ. The particular form of the functions f_θ and h_θ must be carefully chosen to maximize expressiveness while preserving desirable properties for optimization, but from the perspective of estimation and control, their exact form is unimportant.

The RNN is trained on the dataset $\mathcal{D}$ to minimize a regression loss on the input-output map over complete sequences:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \sum_{j=1}^{N} \sum_{k=1}^{K_j} \|\bar{y}[j,k] - h_\theta(x[j,k])\|_2^2 \\ \text{subject to} \quad & x[j,0] = \mathbf{0} \\ & x[j,k+1] = f_\theta(x[j,k], \bar{u}[j,k]), \end{aligned} \tag{4}$$

where K_j is the length of the j^{th} trajectory. The fixed initial state of $\mathbf{0}$ is justified in our setting because each trajectory in $\mathcal{D}$ begins at the rest state $\bar{x}_0$. The optimization problem (4) is typically solved with stochastic gradient descent (SGD) using backpropagation, so the gradient of the output prediction loss is allowed to flow through the recursive applications of f_θ. This allows the RNN to learn dynamics where the effect of an input does not appear until many time steps later. RNNs have achieved state-of-the-art results on many sequence modeling tasks, even though the objective is nonconvex and SGD may converge to suboptimal local minima [29].

C. Model-predictive control with reduced-order model

After finding a value of the RNN parameter θ^* that approximately optimizes the learning objective (4), we use the RNN model in a model-predictive control (MPC) framework to optimize the trajectory-tracking objective (2) in an online manner. Assume temporarily (we will return to this assumption in Section III-D) that at time step k we know a value $\hat{x}[k]$ of the abstract RNN state that is consistent with the input and output history from previous time steps

in the real-world system. We solve the short-horizon optimal control problem

$$\begin{aligned}\underset{\substack{\bar{u}[k],\ldots,\\ \bar{u}[k+H-1]}}{\text{minimize}} \quad & \sum_{i=1}^{H} \|z[k+i] - h_{\theta^*}(x[k+i])\|_W^2 \\ & + \alpha \sum_{i=0}^{H-1} d(\bar{u}[k+i-1], \bar{u}[k+i])^2 \\ & + \beta \sum_{i=1}^{H} d(\bar{u}[k+i], I)^2 \qquad (5) \\ \text{subject to} \quad & x[k] = \hat{x}[k], \\ & x[k+i+1] = f_{\theta^*}(x[k+i], \bar{u}[k+i]) \ \forall i.\end{aligned}$$

In the objective (5), $H \ll K$ is the MPC horizon. In the second and third terms, $d : \mathcal{U} \times \mathcal{U} \mapsto \mathbb{R}_{\geq 0}$ is a metric on the input space $\mathcal{U}$. The first regularization term, weighted by constant $\alpha \geq 0$, encourages smoothness of the control signal. The second regularization term, weighted by constant $\beta \geq 0$, encourages the robot to return to its rest state. Solving (5) yields a sequence of inputs $\bar{u}^*[k], \ldots, \bar{u}^*[k+H-1]$ optimized to track the next H steps of the full goal trajectory. Following the standard moving horizon architecture, we apply only the first input $\bar{u}^*[k]$ from the solution to the real system. Then, at time step $k+1$, we solve a new instance of (5). The optimization problem is nonconvex, but the solution from the previous time step provides a high-quality initial guess, so local optimization methods typically perform well as long as the initial guess is not close to a bad local optimum [30]. We obtain an approximate solution with a few steps of gradient descent with momentum. The momentum state from previous MPC steps is persisted and time-shifted in the same manner as the initial guess.

D. Estimating the RNN state

In Section III-C, we assumed the availability of a RNN state $\hat{x}[k]$ that is consistent with previous inputs and outputs from the real-world system. To obtain $\hat{x}[k]$, it is not sufficient to simply evaluate f_{θ^*} recursively on $\bar{u}[1], \ldots, \bar{u}[k-1]$. Because the RNN model is not perfect, the true outputs $\bar{y}[1], \ldots, \bar{y}[k-1]$ obtained from the real-world system may diverge from the outputs $h_{\theta^*}(x[1]), \ldots, h_{\theta^*}(x[k-1])$ predicted by applying the RNN model in this open-loop manner.

Instead we observe that, although the RNN state has no direct physical meaning, the RNN model is still a nonlinear discrete-time dynamical system with known dynamics. Therefore, its state can be estimated from the input-output history using standard techniques from estimation theory. In particular, we apply an extended Kalman filter (EKF) to the RNN model. We now review the well-known EKF equations to highlight the role of the RNN.

The EKF maintains a Gaussian-distributed belief over the RNN state. At time k, the belief is distributed according to

$$x[k] \sim \mathcal{N}(\mu[k], \Sigma[k]),$$

where $\mathcal{N}(\mu, \Sigma)$ is the Gaussian distribution with mean μ and covariance Σ. After sending an input $\bar{u}[k]$ to the system, we then update the belief according to

$$\begin{aligned}\mu[k|k-1] &= f_{\theta^*}(\mu[k-1], \bar{u}[k]), \\ \Sigma[k|k-1] &= F[k]\Sigma[k-1]F[k]^\top + Q[k],\end{aligned} \qquad (6)$$

where

$$F[k] = \frac{\partial f_{\theta^*}}{\partial x}(\mu[k-1],\ \bar{u}[k])$$

is the Jacobian of the RNN dynamics f_{θ^*} with respect to state. The process noise covariance $Q[k] \succeq \mathbf{0}$ represents noise in the RNN abstract state, so it cannot be derived from a physical model or estimated from data. We simply set Q to a scaled identity matrix in this work. Next, the measurement $\bar{y}[k]$ is captured from the real system. We compute the measurement residual $\gamma[k] = \bar{y}[k] - h_{\theta^*}(\mu[k|k-1])$. According to the current belief, $\gamma[k]$ has covariance

$$S[k] = H[k]\Sigma[k|k-1]H[k]^\top + R[k],$$

where

$$H[k] = \frac{\partial h_{\theta^*}}{\partial x}(\mu[k|k-1])$$

is the Jacobian of the RNN observation function h_{θ^*} with respect to state. The sensor noise covariance $R[k] \succ 0$ represents a physically meaningful quantity that can be derived from a model or estimated from data. In this work we use a motion capture system with isotropic noise, so we set R to a constant scaled identity matrix. We then compute the Kalman gain

$$K[k] = \Sigma[k|k-1]H[k]^\top S[k]^{-1}$$

and update the belief according to

$$\begin{aligned}\mu[k|k] &= \mu[k|k-1] + K[k]\gamma[k], \\ \Sigma[k|k] &= (I - K[k]H[k])\Sigma[k|k-1].\end{aligned} \qquad (7)$$

We then use the mean of the belief distribution as the initial state for the MPC problem (5), that is, $\hat{x}[k] = \mu[k]$.

E. Implementation

For the RNN reduced-order dynamics model f_θ, we select the long short-term memory (LSTM) architecture [28]. The LSTM is the *de facto* standard RNN architecture due to its favorable properties for training with gradient descent. Numerical values of the architectural and training hyperparameters are listed in Table I.

We train the RNN in PyTorch [36]. We also solve the model-predictive control problems and implement the EKF in PyTorch due to the ease of differentiating through the RNN model. We run the MPC control loop at $40\,\mathrm{Hz}$.

IV. Experiments

In all experiments, our deformable body is a thin cylinder of uniform closed-cell polyethylene foam, commonly known as a pool noodle, with length $1.5\,\mathrm{m}$ and diameter $6.5\,\mathrm{cm}$. To attach the object to the robot, we press-fit approximately $4\,\mathrm{cm}$ of one end of the pool noodle into a rigid 3D-printed ring attached to the robot end effector. To track the distal end of the object, we attach a rigid assembly of motion capture markers and track its full pose with a Vicon motion capture

TABLE I

VALUES OF USER-CHOSEN CONSTANTS IN OUR EXPERIMENTS.

Q	EKF process covariance	$10^{-6}I$
R	EKF measurement covariance	$10^{-2}I$
H	MPC horizon	25
α	MPC smoothness weight	1.0
β	MPC homing weight	1e-1
—	MPC gradient descent rate	4e-1
—	MPC gradient descent steps	5
—	LSTM layers	1
n	Reduced state dimension	200
—	LSTM SGD steps	1e4
—	LSTM SGD learning rate	1e-3
—	LSTM SGD batch size	10
N	# trajs. in dataset	100

system. The measurement $\bar{y}$ has a calibrated offset from the marker assembly to lie on the center line of the pool noodle. Our experimental setup is shown in Figure 1.

The robot is a Franka Emika Panda. To track the pose commands $\bar{u} \in SE(3)$ output by the MPC controller, we apply a proportional-only control law in both the position and in the attitude quaternion to generate desired linear and angular velocities v for the end effector. We compute the kinematic Jacobian from `libfranka` and produce desired joint velocities using the Jacobian pseudoinverse with null space optimization towards the "home" position [37].

A. Model frequency response

To validate our RNN model, we compare its empirical frequency response near the internal state $x = \mathbf{0}$ to that of the true physical system at its rest state $\bar{x}_0$. Frequency response is a holistic property of the model that depends on its behavior in multiple parts of the state space. To avoid the complications of frequency-domain analysis for multiple-input/multiple-output systems, we actuate the system only in the yaw axis ψ and measure only the deflection of the pool noodle tip in the horizontal axis. Yaw inputs, as opposed to pitch inputs, avoid the asymmetric effect of gravity.

We sample 33 geometrically-spaced frequencies ranging from $1/8\,\mathrm{Hz}$ to $2\,\mathrm{Hz}$. For each frequency, we apply a yaw input with a small amplitude ($11.5°$ peak-to-peak) for $20\,\mathrm{sec}$ and record the trajectory of the pool noodle tip $\bar{y}$. The small input amplitude allows higher-frequency inputs before reaching the robot's actuator limits. We discard approximately the first half of the recording (on the nearest whole cycle boundary) to eliminate transient effects. We then compute the empirical gain and phase by taking the inner product of the output signal with complex exponential functions, analogous to the discrete Fourier transform. Because the input and output have different units, only the relative gain magnitudes between different frequencies are meaningful. We normalize the gains such that the gain of the real system for the slowest input frequency equals 1.

Bode (gain and phase) plots for each system are overlaid in Figure 3. In the true physical system, we observe typical behavior of a resonant lowpass filter with a strong resonant peak around $0.8\,\mathrm{Hz}$. The peak gain is approximately 4 times larger than the low-frequency gain. This represents a dramatic phenomenon that a faithful model must capture. In the RNN model, the gain response closely matches the real system. The phase response matches closely below the resonant frequency, but begins to lag behind the true system for high frequencies. We suspect this may be due to an unbalanced training data distribution. Overall, this experimental result suggests that the RNN model has successfully captured the frequency response of the physical system.

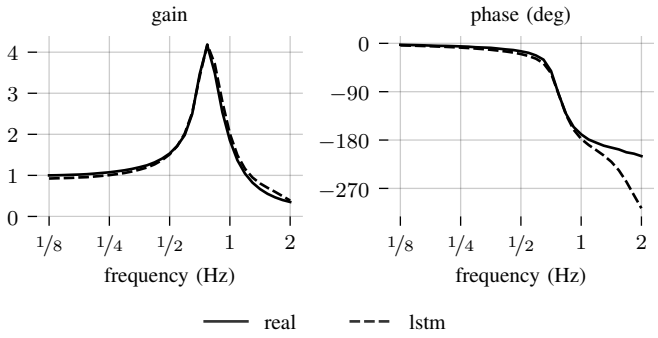

Fig. 3. Frequency-domain gain and phase response (Bode plots) for real pool noodle and LSTM model.

B. MPC tracking

We apply our method to track several test trajectories which attempt to expose the controller's performance with regard to resonant dynamics. The goal trajectories $z[1], \ldots, z[K] \in \mathbb{R}^3$ are specified by the user. As described in eq. (2), our tracking cost is non-isotropic. We use the value $W = \mathrm{diag}(0, 1, 1)$ to focus only on tracking in the Y- and Z-axes, which represents the view from the front of the robot. Without this non-isotropic weight, the goal trajectory would need to be a carefully designed curve in $\mathbb{R}^3$ to be trackable with zero error.

We show the results from three trajectories in Figure 4. The first two trajectories are a circle of diameter $0.6\,\mathrm{m}$ and a figure-8 Lissajous curve of width $1\,\mathrm{m}$ and height $0.4\,\mathrm{m}$. Both trajectories are sinusoidal in each axis, which matches the data collected during our training step. The circle and the vertical axis of the Lissajous curve are set to the system's resonant frequency $0.8\,\mathrm{Hz}$ as determined experimentally. The third trajectory is a rectangle with constant linear velocity along each edge (no stopping at corners). These trajectories are close to the robot's dynamic limits as discussed in Section III-A. Improvements in our low-level controller are required to test more demanding trajectories that push the system further into nonlinearity.

To show that the EKF observer provides meaningful feedback to the controller (the "closed-loop" setup), we compare it to a setup that assumes the predicted feedforward state, i.e. the value yielded by applying the RNN dynamics f_{θ^*} to the full sequence of past inputs, is always correct (the "open-loop" setup). The results of this comparison are visualized in Figures 4 and 5, and the tracking errors are compared numerically in Table II.

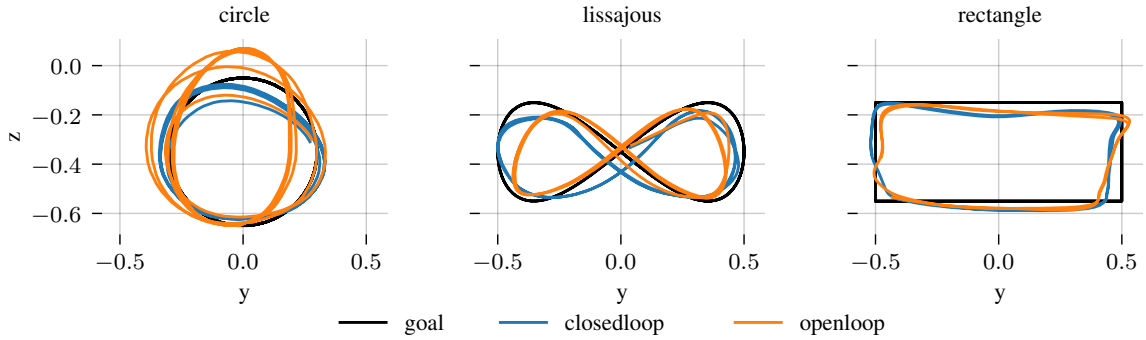

Fig. 4. Two-dimensional projections of paths traced by pool noodle free end in MPC tracking experiments.

TABLE II
MPC TRACKING ERRORS

shape	kind	max error (cm)	mean error (cm)
circle	closedloop	12.59	5.59
	openloop	29.15	11.61
lissajous	closedloop	9.19	4.43
	openloop	14.85	4.77
rectangle	closedloop	14.60	6.34
	openloop	14.62	6.01

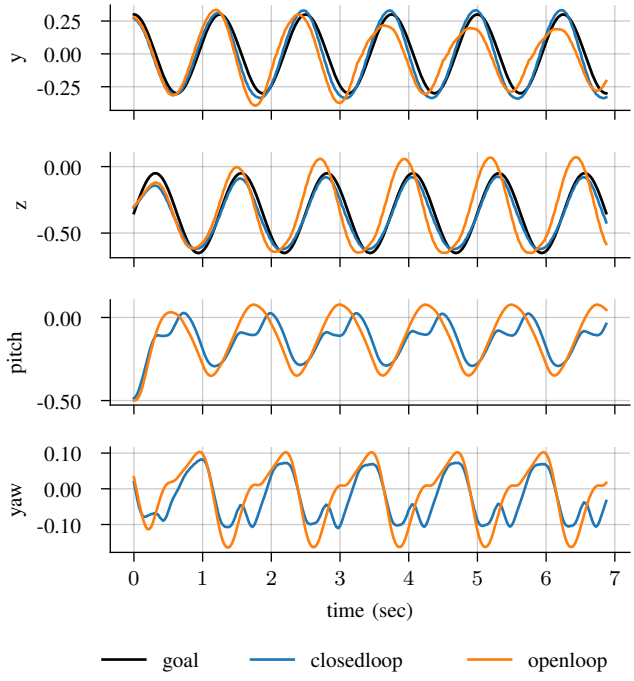

Fig. 5. Traces of rotation angle inputs (pitch, yaw) and horizontal and vertical components of pool noodle free end (y, z) for MPC tracking of the circle trajectory (see Figure 4). The closed-loop variant of our method shows superior tracking performance.

For the circle trajectory, we observe significantly improved performance in both mean and maximum error from the closed-loop setup. In the traces over time, shown in Figure 5, the open-loop solution drifts towards stronger resonance in the vertical axis and weaker resonance in the horizontal axis. In contrast, the error of the closed-loop solution does not grow over time, indicating that our EKF setup is able to compensate for model error. For the Lissajous trajectory, the closed-loop setup yields a minor improvement in mean tracking error but a significant improvement in maximum error. The rectangle trajectory shows little difference between the open- and closed-loop approaches, but it is noteworthy that both approaches track the rectangle reasonably, even though it is physically infeasible and not similar to the training data. This result suggests that the LSTM model behaves reasonably for sequence inputs that are dissimilar to those in the training data.

V. CONCLUSION

We have described and demonstrated a system to manipulate a deformable object such that a particular point on the object tracks a fast trajectory. Our approach is completely data-driven, and requires a fixed initial data-collection phase, without further exploratory actions. We model our dynamical system as an LSTM recurrent neural network and design a nonlinear MPC controller. We use an EKF state observer to account for model error by estimating a value of the LSTM hidden state consistent with past inputs and outputs.

We validate our model on a real hardware setup with a robot manipulator holding a foam pool noodle, measured by a motion capture system. Our experiments show that closing the loop with the EKF observer improves tracking performance compared to an open loop control scheme for several of the test trajectories.

In future work, we aim to improve tracking accuracy by investigating other nonlinear state estimation methods and MPC implementations. The EKF is one of several ways to account for modeling error. Other nonlinear state estimators such as particle filters and moving window least-squares estimators could also be adapted to estimate RNN internal states. Alternatively, a pure deep-learning approach might add past observations as inputs to the RNN. For MPC, a more sophisticated constrained optimization scheme could be applied to the nonlinear MPC problem (5) to help enforce smoothness and actuator limits. We are also interested in applying our work to non-position-based control tasks, such as force control for using deformable objects as tools.

REFERENCES

[1] V. E. Arriola-Rios, P. Guler, F. Ficuciello, D. Kragic, B. Siciliano, and J. L. Wyatt, "Modeling of deformable objects for robotic manipulation: A tutorial and review," *Frontiers in Robotics and AI*, vol. 7, p. 82, 2020.

[2] J. Zhu, A. Cherubini, C. Dune, D. Navarro-Alarcon, F. Alambeigi, D. Berenson, F. Ficuciello, K. Harada, X. Li, J. Pan, and W. Yuan, "Challenges and outlook in robotic manipulation of deformable objects," *CoRR*, vol. abs/2105.01767, 2021.

[3] G. A. Holzapfel, "Nonlinear solid mechanics: A continuum approach for engineering science," *Meccanica*, vol. 37, no. 4, pp. 489–490, Jul. 2002.

[4] P. Wriggers, *Nonlinear Finite Element Methods*. Berlin Heidelberg: Springer-Verlag, 2008.

[5] S. Zimmermann, R. Poranne, and S. Coros, "Dynamic manipulation of deformable objects with implicit integration," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4209–4216, Apr. 2021, conference Name: IEEE Robotics and Automation Letters.

[6] J. M. Bern, P. Banzet, R. Poranne, and S. Coros, "Trajectory optimization for cable-driven soft robot locomotion." in *Robotics: Science and Systems*, vol. 1, 2019, issue: 3.

[7] S. Duenser, J. M. Bern, R. Poranne, and S. Coros, "Interactive robotic manipulation of elastic objects," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3476–3481.

[8] Y. Li, T. Du, K. Wu, J. Xu, and W. Matusik, "DiffCloth: Differentiable cloth simulation with dry frictional contact," *arXiv preprint arXiv:2106.05306*, 2021.

[9] Y.-L. Qiao, J. Liang, V. Koltun, and M. C. Lin, "Scalable differentiable physics for learning and control," *arXiv preprint arXiv:2007.02168*, 2020.

[10] E. Heiden, M. Macklin, Y. Narang, D. Fox, A. Garg, and F. Ramos, "DiSECt: A differentiable simulation engine for autonomous robotic cutting," *arXiv preprint arXiv:2105.12244*, 2021.

[11] J. Barbič and J. Popović, "Real-time control of physically based simulations using gentle forces," *ACM transactions on graphics (TOG)*, vol. 27, no. 5, pp. 1–10, 2008.

[12] M. Thieffry, A. Kruszewski, C. Duriez, and T.-M. Guerra, "Control design for soft robots based on reduced-order model," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 25–32, 2018, publisher: IEEE.

[13] D. Hahn, P. Banzet, J. M. Bern, and S. Coros, "Real2sim: Visco-elastic parameter estimation from dynamic motion," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–13, 2019, publisher: ACM New York, NY, USA.

[14] D. Sulsky, Z. Chen, and H. L. Schreyer, "A particle method for history-dependent materials," *Computer Methods in Applied Mechanics and Engineering*, vol. 118, no. 1, pp. 179–196, Sep. 1994.

[15] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, "ChainQueen: A real-time differentiable physical simulator for soft robotics," *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[16] M. Macklin, M. Müller, and N. Chentanez, "XPBD: position-based simulation of compliant constrained dynamics," in *Proceedings of the 9th International Conference on Motion in Games*, 2016, pp. 49–54.

[17] M. Muller, B. Heidelberger, M. Teschner, and M. Gross, "Meshless deformations based on shape matching," p. 8.

[18] D. Mcconachie and D. Berenson, "Estimating model utility for deformable object manipulation using multiarmed bandit methods," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 967–979, Jul. 2018, conference Name: IEEE Transactions on Automation Science and Engineering.

[19] D. McConachie, A. Dobson, M. Ruan, and D. Berenson, "Manipulating deformable objects by interleaving prediction, planning, and control," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 957–982, Jul. 2020, publisher: SAGE Publications Ltd STM.

[20] J. M. Bern, Y. Schnider, P. Banzet, N. Kumar, and S. Coros, "Soft robot control with a learned differentiable model," in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, May 2020, pp. 417–423.

[21] N. M. Mirza, "Machine learning and soft robotics," in *2020 21st International Arab Conference on Information Technology (ACIT)*, Nov. 2020, pp. 1–5.

[22] D. Bruder, B. Gillespie, C. D. Remy, and R. Vasudevan, "Modeling and control of soft robots using the koopman operator and model predictive control," *arXiv:1902.02827 [cs]*, Jul. 2019, arXiv: 1902.02827.

[23] M. T. Gillespie, C. M. Best, E. C. Townsend, D. Wingate, and M. D. Killpack, "Learning nonlinear dynamic models of soft robots for model predictive control with neural networks," in *2018 IEEE International Conference on Soft Robotics (RoboSoft)*, Apr. 2018, pp. 39–45.

[24] A. P. Sabelhaus and C. Majidi, "Gaussian process dynamics models for soft robots with shape memory actuators," in *2021 IEEE 4th International Conference on Soft Robotics (RoboSoft)*, Apr. 2021, pp. 191–198.

[25] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 124–134, Feb. 2019, conference Name: IEEE Transactions on Robotics.

[26] S. Tonkens, J. Lorenzetti, and M. Pavone, "Soft robot optimal control via reduced order finite element models," *arXiv preprint arXiv:2011.02092*, 2020.

[27] M. Bualat, J. Barlow, T. Fong, C. Provencher, and T. Smith, "Astrobee: Developing a free-flying robot for the international space station," in *AIAA SPACE 2015 Conference and Exposition*. American Institute of Aeronautics and Astronautics, eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2015-4643.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997, publisher: MIT Press.

[29] Z. C. Lipton, "A critical review of recurrent neural networks for sequence learning," *CoRR*, vol. abs/1506.00019, 2015.

[30] F. Allgöwer and A. Zheng, *Nonlinear model predictive control*. Birkhäuser, 2012, vol. 26.

[31] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960, publisher: Citeseer.

[32] O. Nelles, *Nonlinear system identification*. Berlin: Springer, 2001.

[33] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 2455–2467.

[34] E. Terzi, F. Bonassi, M. Farina, and R. Scattolini, "Learning model predictive control with long short-term memory networks," *International Journal of Robust and Nonlinear Control*, vol. Early Access, no. n/a, 2021.

[35] S. L. Brunton and J. N. Kutz, *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[37] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, ser. Advanced Textbooks in Control and Signal Processing. London: Springer-Verlag, 2009.