Automatically-Tuned Model Predictive Control for an Underwater Soft Robotic Arm

David Null[1,2], Teodor Tchalakov[1], William Edwards[3], Kris Hauser[1,3*], and Y Z [1,2,4*]

I. Introduction

Soft robotics is a highly active field of research in robotics due to their potential to comply safely to disturbances and conform shapes to objects and obstacles, in analogy to biological systems like elephant trunks and octopus arms [6]. Soft robot arms are especially appealing candidates for underwater operations thanks to their actuation mechanisms. However, modeling and controlling soft arms to achieve precision tasks has remained a significant challenge. Because soft actuators are nonlinear, have infinite degrees of freedom, and exhibit complex dynamic behavior such as oscillation and hysteresis, analytical models can be incomplete or too computationally intensive for real time operations. Machine learning approaches, such as neural networks, have been widely explored to model the soft robot kinematics and dynamics from observations [7], [1]. Such models can be trained using supervised learning or reinforcement learning (RL) [8], but achieving good generalization across tasks remains a challenge.

In the context of an underwater soft robotic arm, this paper demonstrates that learned model predictive control (MPC) approaches, which model the dynamics of the system, are able to achieve high positioning accuracy. Learned MPC has indeed been applied to soft robot control in prior work [5], but we address the problem that MPC is highly sensitive to the hyperparameters of the learned model and optimizer, and tuning them manually is a tedious process. In this work, we adopt the AutoMPC [2] approach to automatically tune the MPC controller from an offline dataset, and extend it to handle multi-task tuning. On a two-module, hydraulically-actuated underwater soft robot, we compare this method to a learned inverse kinematics model that predicts the actuation pressures needed to achieve a static configuration. Although both controllers were trained on the same dataset, AutoMPC achieved 75% improvement in targeting accuracy.

II. Underwater Soft Robotic Arm

Our experimental platform is a two-module planar underwater soft robotic arm, driven by four hydraulic actuators: two for each module. A pump in the same water tank as the robot is the pressure source for the actuators. The centerline of the robot is marked with color-based fiducials for computer vision tracking. The robot is submerged in approximately 3.5 cm of water so that the computer vision markers remain above the water level. A calibrated camera setup tracks the markers to provide (x,y) coordinates of the center of each section of the arm. Each hydraulic actuator is equipped with two solenoid valves and a pressure sensor. The solenoid valves take a binary control signal and allow water to flow in or out of the actuator. Thus, the 26-D robot's state $x = (x_{marker}, x_{pressure})$ consists of eleven Cartesian marker locations and four analog pressure readings, and its control u consists of 8 binary solenoid inputs. The red marker at the base of the robot is designated as the origin and is seen at the bottom of Fig. 1(a). The end effector point x_{ee} is represented by the marker at the top of the robot.

III. Open-Loop Learned Inverse Kinematics Controller

As a performance benchmark, we implemented an open loop controller which uses a learned inverse kinematics model to predict the actuator pressure values needed to produce a given end effector position: $x_P = \hat{f}(x_{ee})$. The model is a multilayer deep neural network with inputs corresponding to the (x, y) coordinates of the desired end effector marker location and outputs corresponding to the four pressure values for each actuator. Training data was collected by running the robot through a series of planned and random trajectories. The coverage of the 11,358 data samples is shown in Fig. 1(b). The control algorithm first generates 4 desired pressure values x_P^d from a given target location using the learned model, and then drives each actuator to the desired pressure simultaneously. Once a pressure target it reached, both valves are closed and the actuator is locked. Although we use feedback to correctly pressurize the actuators, the controller is open loop with regard to x_{ee}.

IV. AutoMPC Controller

Data-driven model predictive control (MPC) consists of a learned dynamics model provided as input to a finite-horizon trajectory optimizer. A benefit of MPC over RL-based methods is that it can be easily customized to new tasks by changing the objective function used in the optimizer. The dynamics can also be learned from data collected offline in a controlled and safe fashion, rather than allowing the robot to generate uncontrolled inputs that may risk damaging the robot. However, closed-loop performance of MPC is

*Corresponding authors: K.H. kkhauser@illinois.edu and Y Z zhyang@illinois.edu

[1]Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

[2]Beckman Institute for Advanced Science and Technology, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

[3]Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

[4]Department of Nuclear, Plasma, and Radiological Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

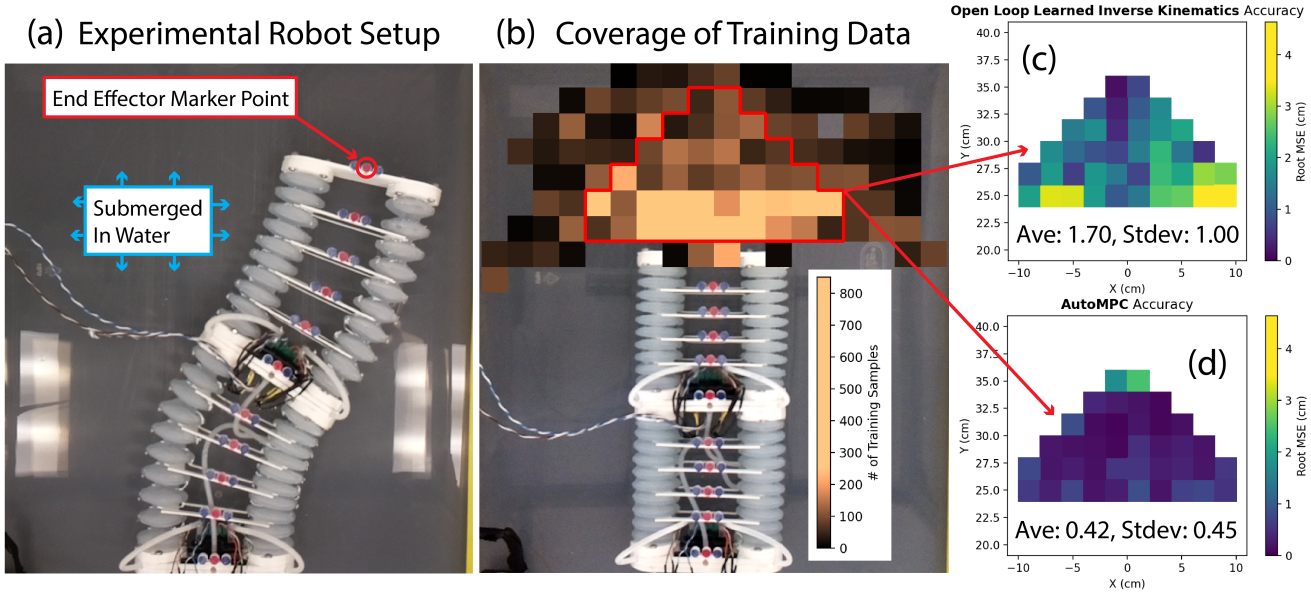

Fig. 1. (a) Experimental setup of 2D underwater soft robotic arm. (b) Coverage map of training data used to build controllers. (c) Accuracy map of 'Open-Loop Learned Inverse Kinematics Controller'. (d) Accuracy map of 'AutoMPC Controller.'

known to be sensitive to the model hyperparameters, horizon length, optimizer parameters, and regularization terms in the objective function [4]. We build on the open-source AutoMPC software package [2], which uses Bayesian optimization techniques from the automatic machine learning (AutoML) community to tune hyperparameters [3]. In this work we extended the AutoMPC software to handle binary outputs and to tune controllers for performance across a set of multiple tasks, and we have made these extensions available for other researchers.

Standard trajectory optimization cannot be directly applied to our robot because the solenoid controls are binary on/off signals whereas typical trajectory optimizers output continuous control signals. Hence, we need to modify the optimizer output. Our approach is to postprocess the controls output by trajectory optimization to round to the nearest value (0 or 1). Although this is a somewhat naïve approach that introduces errors in the internal MPC predictions, the tuning process mitigates the emergent effects of this prediction error.

To train our system, AutoMPC learns the dynamic behavior of the arm in the standard form $x[t+1] = f(x[t], u[t])$. We defined a task as reaching a desired end effector position x_{ee}^d. The performance objective was defined as 1 if the end effector lies outside a 2x2 cm^2 square centered on the target x_{ee}^d, and 0 if lies inside the location (low is better). We sample a set of tasks from the reachable workspace, and AutoMPC tunes the controller by minimizing the average of the performance measure across tasks, using closed-loop evaluation on a surrogate dynamics function learned from the same dataset. Ultimately, AutoMPC learned a 3-layer perceptron (MLP) model with 227 hidden units, ReLU nonlinear layer, and an MPC horizon of 23. This MPC configuration achieved over twice the accuracy of the default configuration.

V. Results

To compare the two controllers, forty end effector target locations were chosen for the robot to attempt to reach starting from a fully depressurized home position shown in Fig. 1(b). In each run, the controller was given a maximum of 50 seconds to stabilize the robot at the target location. The root mean squared errors (RMSE) achieved by both controllers at each of the forty target locations are shown in Fig. 1(c) and 1(d). The AutoMPC controller obtained a smaller RMSE for nearly all target locations. It was also more consistent, achieving a smaller standard deviation of error.

The task of manipulation is impacted by the accuracy of the arm which holds the manipulator. We want to explore if the benefits that come from using compliant materials in the design of robotic grippers can be realized by a robotic arm. Underwater environments can reduce the challenges faced when controlling soft actuators. In fact, living organisms that live underwater primarily use compliant manipulators. This work presents the results of using two different control strategies to actuate an underwater soft robotic arm to chosen target locations.

In future work we are interested in extending the work to tasks like grasping, and extending to longer robot arms composed of more modules. We also would like to relax the assumption of full vision feedback on the robot's spine, and to investigate how an AutoMPC controller can work in the partially-observed case.

References

[1] K. Chin, T. Hellebrekers, and C. Majidi, "Machine Learning for Soft Robotic Sensing and Control," *Advanced Intelligent Systems*, vol. 2, no. 6, p. 1900171, 2020.

[2] W. Edwards, G. Tang, G. Mamakoukas, T. Murphey, and K. Hauser, "Automatic tuning for data-driven model predictive control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[3] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Neural Information Processing Systems (NeurIPS)*, 2015, pp. 2962–2970.

[4] M. Forgione, D. Piga, and A. Bemporad, "Efficient calibration of embedded MPC," in *IFAC World Congress*, 2020.

[5] 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," *2018 IEEE International Conference on Soft Robotics (RoboSoft)*, pp. 39–45, 2018.

[6] J. Hughes, U. Culha, F. Giardina, F. Guenther, A. Rosendo, and F. Iida, "Soft Manipulators and Grippers: A Review," *Frontiers in Robotics and AI*, vol. 3, p. 69, 2016.

[7] D. Kim, S.-H. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee, J. Bae, Y.-L. Park, K.-J. Cho, and S. Jo, "Review of machine learning methods in soft robotics," *PLOS ONE*, vol. 16, no. 2, p. e0246102, 2021.

[8] X. You, Y. Zhang, X. Chen, X. Liu, Z. Wang, H. Jiang, and X. Chen, "Model-Free Control for Soft Manipulators Based on Reinforcement Learning," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2909–2915, 2017.