

# 1 Robot Prototype

2 The robot, as depicted in Figure 1, consists of a flex-  
 3 ible backbone rigidly affixed to spacers, accompa-  
 4 igned by four rods fixed at the end spacer and passing  
 5 through the remaining spacers with sufficient clear-  
 6 ance, forming the primary body of the robot.

7 To drive the robot, four brushless DC motors from  
 8 Maxon Motors, equipped with quadratic encoders  
 9 and 150:1 reduction gearheads, are utilized. Precise  
 10 motor position control is achieved through four PID  
 11 position controller modules (EPOS4 Compact 50/5  
 12 CAN), which receive encoder feedback and commu-  
 13 nicate with a PC using the CAN protocol to establish  
 14 and retrieve controller set-points and configurations.  
 15 Lead screws, connected to braided tubes via 3D  
 16 printed connectors, are attached to the motors to con-  
 17 vert motor power into tube-pulling and pushing ac-  
 18 tions. A schematic of the robot is shown in Figure 2.

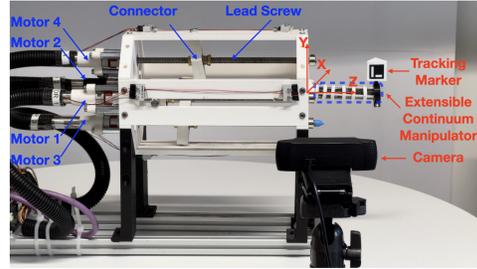


Figure 1: Prototype of the flexible robotic arm composed of a reinforced multi-backbone robot. The robot is connected to four brushless DC motors using lead screws. An ArUco marker [1, 2] is placed on the robot tip, and a camera is used to track the marker’s position.

# 19 2 Network Architecture and Training

20 Table 1 presents a summary of the hyperparameters and  
 21 network structure. It should be noted that we employed an  
 22 early-stopping technique to prevent overfitting when training  
 23 the model. With early stopping, the model’s training is  
 24 halted before it starts to overfit the training data, even if all  
 25 iterations or epochs have not been completed. This allows  
 26 the model to avoid memorizing the training data excessively  
 27 and improves its ability to generalize to new, unseen data.

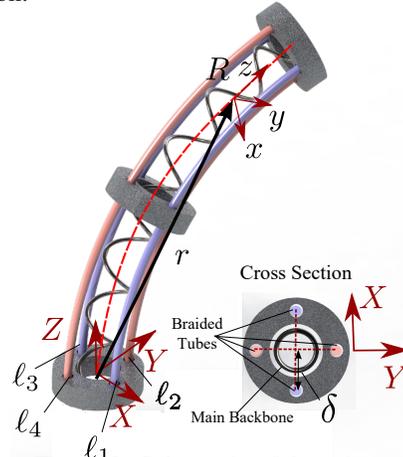


Figure 2: Schematic of the robot.

Table 1: Hyperparameters and network structure.

Hyperparameter	value
No. of hidden neuron ( $\theta$ )	112 (64,32,16)
Augmented vector size (p)	64
No. of hidden layers	3
Activation functions	ELU
Learning rate	0.001
Type of ode-solver	fixed-adams
Absolute tolerance for ode-solver	1e-9
Relative tolerance for ode-solver	1e-7
Number of iteration	7000

# 28 3 Controller Configuration

29 This section will provide the details of the controller configurations including its hyperparameters,  
 30 running cost, and terminal cost functions.

31 The dynamics of the controlled system is captured by the trained FK model (augmented neural ODE  
 32 model), while the running cost and terminal state cost are defined as follows:

- **Running cost:** our running cost function is composed of three costs and defined as follows:

$$\begin{aligned}
\text{cost\_tracking} &= w_{\text{tracking}} \cdot \|\mathbf{x} - \mathbf{x}_{\text{reference}}\|^2 \\
\text{cost\_obstacles} &= w_{\text{obstacle}} \cdot (d_1 < 0.01) + (d_2 < 0.01) \\
\text{cost\_jerk} &= w_{\text{jerk}} \cdot \|\mathbf{u} - \mathbf{u}_{\text{previous}}\|^2 \\
\text{cost\_affordance} &= w_{\text{affordance}} \cdot \text{affordance\_measure} \\
\text{running\_cost} &= \text{cost\_tracking} + \text{cost\_obstacle} + \text{cost\_jerk} + \text{cost\_affordance}
\end{aligned}$$

34 where  $\mathbf{x}$  represents the current state of the system,  $\mathbf{x}_{\text{reference}}$  is the corresponding state in  
35 the reference trajectory,  $\mathbf{u}$  denotes the current control input, and  $\mathbf{u}_{\text{previous}}$  represents the  
36 previous control input. The weights  $w_{\text{tracking}}$ ,  $w_{\text{obstacle}}$ , and  $w_{\text{jerk}}$  control the importance of  
37 each term in the overall cost function.  $w_{\text{affordance}}$  determines a suitable metric or measure  
38 that quantifies the affordance for the given task or goal. The first term penalizes the deviation  
39 of the reference trajectory. These deviations are weighted by a factor of 200, encouraging  
40 the system to closely follow the desired trajectory. The second term is a penalty term that  
41 considers the distance between the current states and two obstacle locations, denoted as  $d_1$   
42 and  $d_2$ . If the distance to either obstacle is less than 0.01, a high penalty of 100,000 is added.  
43 This incentivizes the system to avoid approaching the obstacles too closely. To discourage  
44 jerky and abrupt movements, we considered another penalty term. This term penalizes high  
45 rates of change in acceleration or control inputs. In our implementation,  $w_{\text{jerk}}$  is set to 0.1.

46 • **Terminal cost:** our terminal cost is defined as:  $\text{terminal\_cost} = w_{\text{terminal}} \cdot \|\mathbf{x} - \mathbf{x}_{\text{goal}}\|^2$ ,  
47 where  $w_{\text{terminal}}$  is the weighting factor that controls the importance of the terminal cost.

48 The  $\lambda$  parameter was set to 1 to balance the importance between the running cost and terminal  
49 state cost. The control inputs were constrained within the range defined by  $\text{umin} = [-0.01, -0.01, -$   
50  $0.01]$  and  $\text{umax} = [0.01, 0.01, 0.01]$ . Gaussian noise with a standard deviation of  $\text{noise\_sigma} =$   
51  $0.001 * \text{torch.eye}(3)$  was added to control samples for exploration. The MPPI optimization process  
52 involved generating 500 control samples per iteration, with a prediction horizon of 10 time steps.  
53 These parameter values were chosen to achieve effective control performance and can be fine-tuned  
54 for specific application requirements.

## 55 4 Affordance

56 In the context of robotics, an affordance is a relationship between an actor (i.e., robot), an action  
57 performed by the actor, an object on which this action is performed, and the observed effect [3].  
58 The general idea of the affordance theory can be used in robotics to provide some information of  
59 mapping between objects, agents and the actions they can take on each other, as there is no unified  
60 formalization of it in robotics.

61 In our implementation, we incorporate a set of affordance terms (penalties for violating the motion  
62 restrictions) into the running cost of the controllers which can be selectively activated or deactivated  
63 by the operator, depending on the task phase. Thanks to the versatility of MPPI, which can handle  
64 non-convex running costs, allows us to effectively utilize these affordance terms for a more intuitive  
65 and context-aware interaction between the operator, the robot, and the environment, enabling more  
66 effective and efficient teleoperation. By adding the affordance measure to the running cost, we give  
67 more weight to actions that align with the desired affordance.

## 68 References

69 [1] S. Garrido-Jurado, R. M. noz Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez. Automatic  
70 generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*,  
71 47(6):2280 – 2292, 2014. ISSN 0031-3203. doi:<http://dx.doi.org/10.1016/j.patcog.2014.01.005>.  
72 URL <http://www.sciencedirect.com/science/article/pii/S0031320314000235>.

- 73 [2] S. Garrido-Jurado, R. M. noz Salinas, F. Madrid-Cuevas, and R. Medina-Carnicer. Generation of  
74 fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, 51:  
75 481 – 491, 2016. ISSN 0031-3203. doi:<http://dx.doi.org/10.1016/j.patcog.2015.09.023>. URL  
76 <http://www.sciencedirect.com/science/article/pii/S0031320315003544>.
- 77 [3] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances:  
78 from sensory–motor coordination to imitation. *IEEE Transactions on Robotics*, 24(1):15–26,  
79 2008.