MAPLES: Model based Assistive Policy Learning for Shared-autonomy

Rolif Lima*, Somdeb Saha, and Kaushik Das

Abstract— **This paper presents a novel approach for shared autonomy in robot teleoperation that utilises a policy adaptation technique. This is achieved by modelling shared autonomy as an optimal control problem with a cost function that combines operator reference tracking and goal tracking costs. A neural network policy is trained from this optimal control problem and the resulting Hamiltonian as the loss function. The state error and goal error are fed as inputs to this policy, which outputs velocity commands for the robot's end-effector. The resulting learnt policy is invariant to scaling, translation, and rotation. Preliminary experiments validate the robustness of the learned policy in tracking the operator's inputs while providing assistance to reach the goal accurately. The policy generalises well to trajectories with noise, novel input trajectories, and scenarios with closely spaced goals. Compared to conventional model predictive control solvers, the learned policy offers significant computational speedup, enabling real-time shared autonomy. The proposed approach paves the way for efficient shared autonomy in continuous teleoperation tasks with assistive goal reaching.**

I. Introduction & Related Work

Teleoperation systems have been leveraged for their ability to provide safe access to remote environments and have found their use in applications such as surgical robots, hazardous waste handling, deep ocean, and space exploration [1], [2], [3], [4]. Such systems usually consist of a remotely located robot controlled by a human operator using a locally situated control system connected over a communication network. The local control system allows the operator to communicate to the robot via different input interface such as language, physical human-robot interaction, AR/VR, haptic teleoperation devices, and their combination.

Shared Autonomy (SA), also knows as Assistive Control, or Shared Control [5] in context of teleoperation, typically involves a team consisting of a human operator and an autonomous robot. This synergy leverages the unique strengths of both parties, combining human cognitive and decision-making abilities with the precision, strength, and endurance of robots. SA employs algorithms for human-robot interaction that enable seamless cooperation towards achieving a common goal.

Shared Autonomy can broadly be divided into two categories, namely policy blending approach [6], [5], [7] and policy adaptation approach [8], [9]. Policy blending involves predicting the operator's intent and blending the autonomous input with operator's inputs based on the estimated confidence in the intent, often using Bayesian inference or Inverse Reinforcement Learning. However, these methods have limitations like complete transfer of control, instability, and potential catastrophic failure.

Policy adaptation techniques address these issues by directly modifying the operator's inputs based on user's and environment's states. These approaches commonly use Reinforcement Learning with composite reward functions incorporating system stabilisation and task-specific goals[8]. Various methods like DDQN[10], residual-policy learning[11], and hierarchical RL have been proposed for policy adaptation[12], seamlessly incorporating the user's feedback.

Existing shared autonomy techniques rely on precise dynamic models and pre-trained policies, limiting their applicability to fixed number of goals tasks with discrete action spaces. Reinforcement learning (RL) algorithms also face data inefficiency and trust issues when transferring to real-world scenarios. In this paper, we address these gaps by modelling the policy adaptation approach of shared autonomy as an optimal control problem. This enables policy adaptation for continuous action spaces and multiple goals while maintaining a balance of human-robot trust. Additionally, we learn the assistive policy from the proposed optimal control problem, which provides the added benefit of computational efficiency through the learned policy.

Main contribution of this work are listed below:

- Shared autonomy problem is modelled as an optimal control problem.
- A framework for learning an assistive policy is presented to learn from the proposed optimal control problem.
- Assistive policy invariant to scaling, translation and rotation is developed using error based policy model.

The rest of the paper is structured as follows: Section II briefly describes the problem at hand, followed by Section III describing the optimal control formulation which comprises of the cost function, system model and constraints. It also describes the neural network policy along with it's architecture, loss function and training procedure. Section IV provides a brief description of experimental setup followed by results that validate our approach. In Section V we summarise our key findings and their implications along with limitations and future research directions.

II. Problem Description

The experimental setup, illustrated in Figure 1, involves an operator using teleoperation to control a 6 Degrees of Freedom (DOF) Universal Robots UR5 robot with the aid of an HTC Vive Virtual Reality system. The specific task entails the operator picking up blocks from a table in a predefined

*Corresponding author. All authors are with TCS Research, Tata Consultancy Services, Bengaluru, Karnataka - 560066, India. e-mail: {rolif.lima}@tcs.com

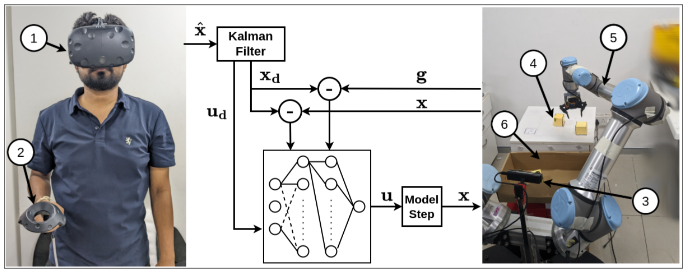

Fig. 1: System: Left shows an operator wearing HMD (1) along with hand held controller(2); Right shows stereo camera (3), goals (4), robot arm (5) and drop box (6), middle shows the learned policy architecture using data from operator and controlling the robot

sequence and placing them into a designated box. In this study, shared autonomy is formulated as an optimal control problem (OCP) within a policy adaptation framework. This problem is approached and solved using Model Predictive Control (MPC) methodology. The corresponding control policy is learned from the data generated during the experiments. For the purpose of this work, it is assumed that the operator's movements are always directed towards the intended goal, with no abrupt changes in the goal while pursuing a specified one.

III. Optimal Control Formulation

The optimal control problem for achieving operator tracking and assistance in reaching a goal is formulated as follows.

A. Cost function

The cost for the OCP is defined using three components: a reference tracking cost to synchronize the robot's motion with the operator, a goal-tracking cost for autonomous assistance as the robot approaches the goal and finally the operator's input tracking cost to ensure operator gets the feeling of authority over the robot. The transition between reference tracking and the goal tracking is managed by a scalar weight, which is computed based on the user's predicted trajectory and goal locations.

$$\begin{aligned}\min_{\mathbf{x},\mathbf{u}} J(\mathbf{x},\mathbf{u};\mathbf{g}) = \sum_{i}^{N} (& w_{min}(\mathbf{x_{di}},\mathbf{g})||(\mathbf{x_i}-\mathbf{x_{di}})||_{\mathbf{Q}} \\ &+ (1-w_{min}(\mathbf{x_{di}},\mathbf{g}))||{}^{\mathbf{g}}\mathbf{R_b}(\mathbf{x_i}-\mathbf{g}[\mathbf{w_{min}}])||_{\mathbf{P}}) \\ &+ ||(\mathbf{u_i}-\mathbf{u_{di}})||_{\mathbf{R}}\end{aligned} \tag{1}$$

where, the subscript i denotes the i^{th} time step for state $\mathbf{x}$ and control $\mathbf{u}$. $\mathbf{Q} \geq 0$, $\mathbf{R} > 0$ and $\mathbf{P} \geq 0$ are the respective weight matrices. The notation $||(\cdot)||_{(*)}$ is used to represent the quadratic form, i.e., $||(\mathbf{x}-\mathbf{x_d})||_{\mathbf{Q}}$ corresponds to $(\mathbf{x}-\mathbf{x_d})^T\mathbf{Q}(\mathbf{x}-\mathbf{x_d})$. In the goal tracking part, the relative position between the robot's end-effector and goal is transformed to the goal frame via rotation matrix ${}^{\mathbf{g}}\mathbf{R_b}$ and scaled via diagonal matrix $\mathbf{P}$ to achieve the desired goal approach behaviour. The weighing function $w(\mathbf{x_d},\mathbf{g})$ smoothly transitions from tracking operator's commands to assisting in reaching a specific goal state based on the operator's state as given in the eq (2), with $w_{min}(\mathbf{x_d},\mathbf{g})$ being the minimum computed weight corresponding to different goals from the goal set $\mathscr{G}$.

$$w(\mathbf{x_d},\mathbf{g}) = \frac{1}{1+\exp(\beta-\alpha||(\mathbf{x_d}-\mathbf{g})||)} \tag{2}$$

where, α and β are tuning parameters. Since we are only concerned with position of robot's end effector, it is used as state $\mathbf{x}$. The goal $\mathbf{g}$ represents a position in the robot's workspace, and the reference $\mathbf{x_d}$ is generated by forward propagation of a constant acceleration model as $\mathbf{x_d}_{t+1} = \mathbf{x_d}_t + \mathbf{v_d}_t * \delta t + 1/2\mathbf{a_d}_t * \delta t^2$, where, $\mathbf{x_d}_t, \mathbf{v_d}_t$ and $\mathbf{a_d}_t$ are estimated using a Kalman filter employing constant acceleration model for state prediction and operator's measured position as measurement.

B. System model & Constraints

A point mass model is used to represent the motion model for the robot. This flexibility is allowed due to the use of a commercially available 6-DoF manipulator equipped with a high gain inner loop controller for controlling the robot's joints [13]. Consequently, the system behaves similar to an identity-like system to the outer loop optimal controller.

$$\mathbf{x}_{\dot{t+1}} = \mathbf{x}_t + \mathbf{u_t} * \delta t \tag{3}$$

where, $\mathbf{x}_t$ and $\mathbf{u_t}$ corresponds to the position and applied control action (applied velocity) of the robot's end-effector at discrete times step t. Additionally, constraints to avoid the robot from colliding with the environment such as planner and ellipsoidal constraints, and can be used such as in [14], [15].

C. Learning assistive policy

Above optimal control problem needs to be solved at every time step to optimally track operator's motion while simultaneously providing assistance, which is computationally expensive and time-consuming. Therefore we propose to learn a policy as a neural network function approximation guided by the optimal control problem. From optimal control theory, it is known that the the optimal control $\mathbf{u}^*$ minimizes the control Hamiltonian[16]. Thus the Hamiltonian is utilized as the loss function to learn the mapping from states to control action for the robot, as demonstrated in the MPC-Net framework [17].

The control Hamiltonian for the above optimal control problem is derived as given below

$$\begin{aligned}H(\mathbf{x},\pi_\theta(\mathbf{x},\mathbf{x}_d,\mathbf{u}_d,\mathbf{g}),\lambda) = (& w_{min}(\mathbf{x_{di}},\mathbf{g})||(\mathbf{x}-\mathbf{x_d})||_{\mathbf{Q}} \\ &+ (1-w_{min}(\mathbf{x_{di}},\mathbf{g}))||{}^{\mathbf{g}}\mathbf{R_b}(\mathbf{x}-\mathbf{g_j})||_{\mathbf{P}}) \\ &+ ||(\mathbf{u}-\mathbf{u_d})||_{\mathbf{R}} + \lambda^T\mathbf{c} \\ &+ \delta_x V(\mathbf{x})^T \mathbf{f}(\mathbf{x},\pi_\theta(\mathbf{x},\mathbf{x}_d,\mathbf{u}_d,\mathbf{g}))\end{aligned} \tag{4}$$

where, $\pi_\theta(\mathbf{x},\mathbf{x_d},\mathbf{u_d},\mathbf{g})$ represents the parameterised neural network policy, λ represents the vector of Lagrangian multiplier multiplying with the vector of constraints $\mathbf{c}$ and $\delta_x V(\mathbf{x})$

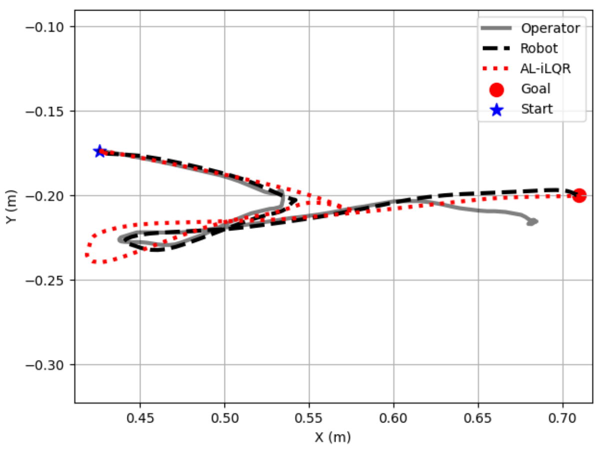

Fig. 2: X-Y plot comparing the trajectories corresponding to operator's commanded reference against learned neural network policy and AL-iLQR

is the gradient of the value function with respect to the state. The optimal parameters θ^* for the policy are computed by minimising the expectation of the Hamiltonian over the states space

$$\theta^* = argmin_\theta E_{\mathbf{x}}[H(\mathbf{x}, \pi_\theta(\mathbf{x}, \mathbf{x}_d, \mathbf{u}_d, \mathbf{g}), \lambda)] \tag{5}$$

D. Policy training

Data corresponding to $\{\mathbf{x}, \mathbf{x_d}, \mathbf{u_d}, \mathbf{g}, \delta_x V(\mathbf{x}), \lambda\}$ at every time step and state is required for learning the policy over the entire state space using the Hamiltonian loss. This is impractical for teleoperation tasks and may also require a large model with large number of parameters. To overcome this, the MPC methodology is used to collect data corresponding to high-probability regions of the operator's state space. Specifically, the AL-iLQR formulation which provides $\delta_x V(\mathbf{x})$ and λ as the by-products in addition to the optimal control and state trajectory is used [18].

Additionally, for better generalisation and reduced input dimension of the policy network, we use error signal representation as input to the policy, with state error defined as $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x_d}$ and goal error defined as $\tilde{\mathbf{g}} = \mathbf{x} - \mathbf{g}$ as inputs, instead of their explicit counterparts which modifies the final Hamiltonian as

$$\begin{aligned} H(\tilde{\mathbf{x}}, \pi_\theta(\tilde{\mathbf{x}}, \mathbf{u}_d, \tilde{\mathbf{g}}), \lambda) = {} & w_{min}(\mathbf{x_{di}}, \mathbf{g})||\tilde{\mathbf{x}}||_{\mathbf{Q}} \\ & + (1 - w_{min}(\mathbf{x_{di}}, \mathbf{g}))||^{\mathbf{g}}\mathbf{R_b}\tilde{\mathbf{g}}||_{\mathbf{P}} \\ & + ||(\mathbf{u} - \mathbf{u_d})||_{\mathbf{R}} + \lambda^T \mathbf{c} \\ & + \delta_x V(\tilde{\mathbf{x}})^T \mathbf{f}(\mathbf{x}, \pi_\theta(\tilde{\mathbf{x}}, \mathbf{u}_d, \tilde{\mathbf{x}})) \end{aligned} \tag{6}$$

E. Data collection

Data for training the policy is generated in two stages, in the first stage, a pair of operator's position trajectories and corresponding goal $(\mathbf{x}_{0:T}, \mathbf{g})$ are collected by teleoperating the robot and stored, additional trajectory samples are generated by augmenting the collected trajectories with scaling and rotation of collected trajectories about the goal

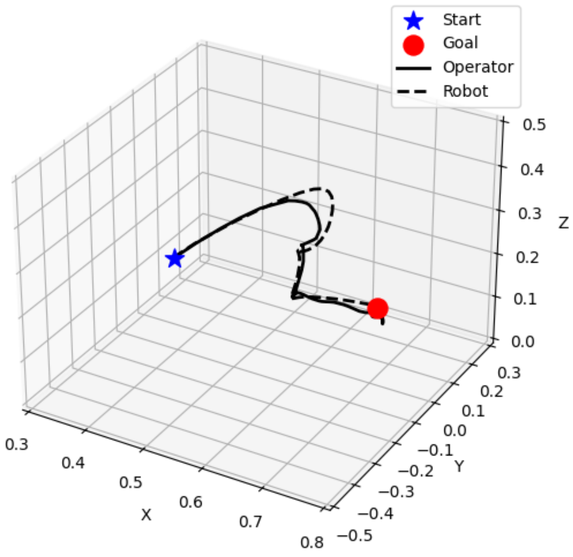

Fig. 3: 3D Trajectory of robot and the commanded operator trajectory

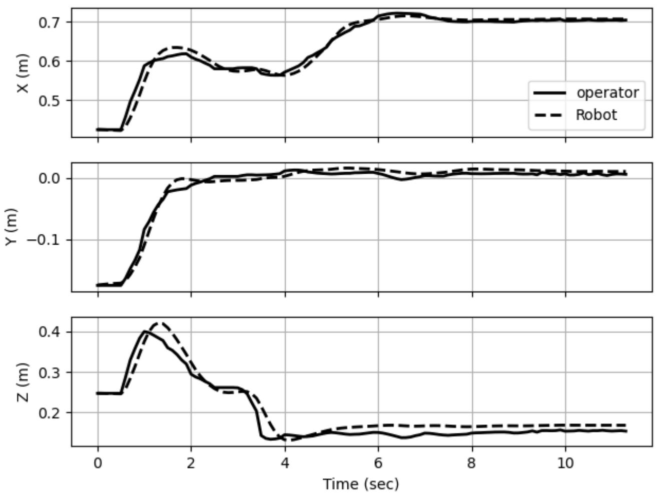

Fig. 4: Variation in X-,Y-,Z-coordinates of operator's and robot's position with respect to time

position. In the second stage, the collected trajectory-goal pairs are passed through to AL-iLQR algorithm to compute the optimal state $\mathbf{x}$, $\delta_x V(\mathbf{x})$ and λ for each input desired state $\mathbf{x_d}$ and goal $\mathbf{g}$. Generated data is further stored in a buffer as $\{\tilde{\mathbf{x}}, \mathbf{u_d}, \tilde{\mathbf{g}}, \delta_x V(\mathbf{x}), \lambda\}$, which is sampled randomly and used for training the policy.

Even with augmented data samples, that cover a large volume of state space, the learnt policy may be biased towards states encountered by the optimal AL-iLQR policy. To counter this, a behaviour policy π_β is used to push the emulated AL-iLQR loop towards the states that will be visited when the learned policy is driving the system

$$\pi_\beta(\mathbf{x}, \theta) = (1 - \alpha)\pi_{ilqr} + \alpha\pi(\mathbf{x}|\theta) \tag{7}$$

where, the mixing parameter α is initially set to zero and linearly increased with the number of iterations upto one

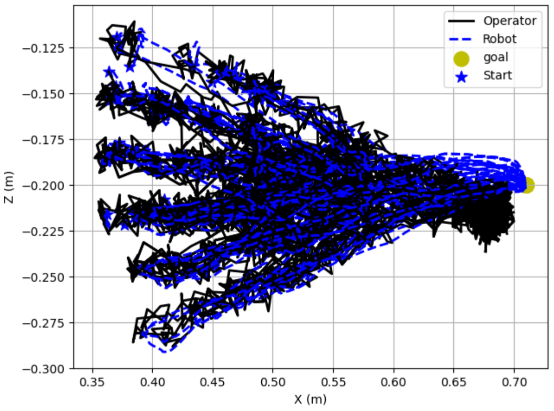

Fig. 5: Evaluation of the learned policy against 64 operators trajectories that were used for training with added random normally distributed noise in them

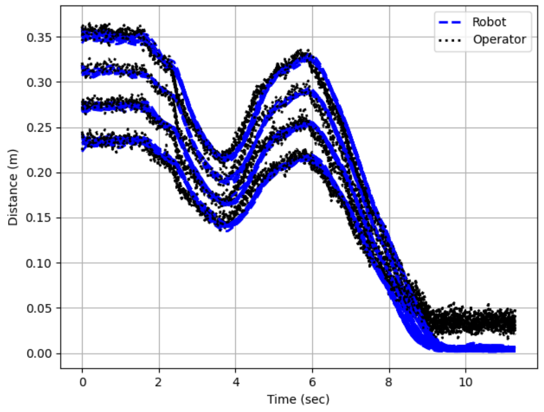

Fig. 6: Variation of distance between the goal and the policy generated robot's trajectory and the operator's trajectory respectively

Algorithm 1 An algorithm for policy training

```
Require: SetOfDemoTraj $D$, AL-iLQRSolver, Policy $\pi_\theta$
Require: ReplayBuffer $\mathbf{B}$,MaxIter, BatchSize, LearningRate
  for Itter in [1:MaxItter] do
      $\alpha \leftarrow 1 - \text{Itter/MaxItter}$
      $\mathbf{g}, \mathbf{x}_{traj} \leftarrow SampleRandomTraj(\text{SetOfDemoTraj})$
      $\mathbf{x}_0 \leftarrow \mathbf{x}_{traj}[0] + \mathcal{N}(0, \sigma)$
      for t in [0:$len(\mathbf{x}_{traj})$] do
          $\mathbf{x}_d, \mathbf{u}_d \leftarrow Kalman(\mathbf{x}_{traj}[t])$
          $\mathbf{x}_{mpc}, \mathbf{u}_{mpc}, \delta_{\mathbf{x}}\mathbf{V}, \lambda \leftarrow \text{AL-iLQR}(\mathbf{x}_0, \mathbf{x}_d, \mathbf{u}_d, \mathbf{g})$
          $\tilde{\mathbf{x}} \leftarrow \mathbf{x} - \mathbf{x}_d$
          $\tilde{\mathbf{g}} \leftarrow \mathbf{x} - \mathbf{g}$
          Append$(\tilde{\mathbf{x}}, \mathbf{u}_d, \tilde{\mathbf{g}}, \lambda, \delta_{\mathbf{x}}\mathbf{V})$ to $\mathbf{B}$
          $\mathbf{x}_0 \leftarrow Step(\mathbf{x_0}, \alpha\mathbf{u}_{mpc} + (1-\alpha)\pi_\theta) + \mathcal{N}(0, \sigma)$
      end for
      if $len(\mathbf{B}) >$ BatchSize then
          $\mathbf{S} \leftarrow$ DrawRandomSamplesBatch($\mathbf{B}$, BatchSize)
          $\mathbf{U} \leftarrow$ EvaluatePolicyOnBatch$(\pi_\theta, \mathbf{S})$
          $l \leftarrow$ ComputeLoss$(\mathbf{U}, \mathbf{S})$
          $\theta \leftarrow$ StepOptimizer$(\delta_\theta l)$
      end if
  end for
```

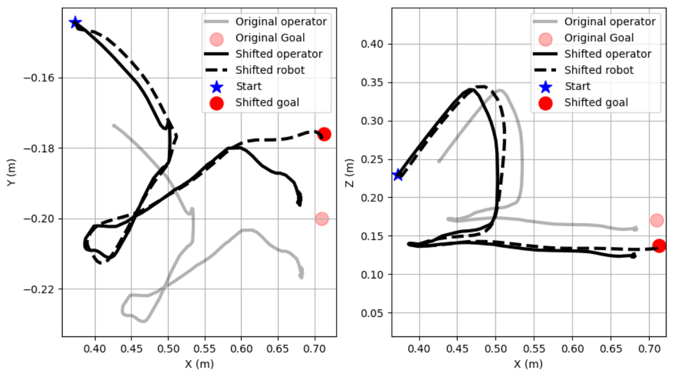

Fig. 7: Out of sample trajectory generated by randomly translation and scaling the original trajectory (indicated with faint plot)

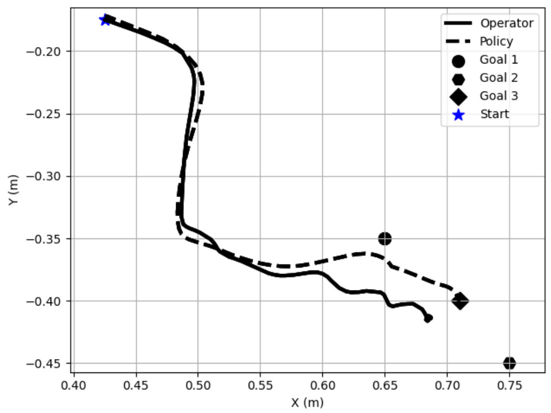

Fig. 8: Robot's and operator's commanded trajectory in the presence of closely located multiple goals

and then maintained as one till the final iteration.

The complete algorithm for training the policy is given in Alg. 1

For the neural network policy, we used a fully connected neural network with 3 layers, with 9 neurons in input layer, 128 neurons in the hidden layer and 3 for output layer. Input for the neural network is formed by concatenating the state error, desired velocity and the goal error as $[\tilde{\mathbf{x}}, \mathbf{u_d}, \tilde{\mathbf{g}}]$. $\tilde{\mathbf{g}}$ is computed by using the goal which is closest to the operator's desired position $\mathbf{x_d}$.

IV. Results

A. Experimental setup:

The experimental setup used is shown in Fig. 1, it comprises of an HTC Vive VR setup with controllers to capture operators motion and HMD to provide the stereoscopic visual feedback, and UR5 robot to perform the manipulation task. We consider a picking task, in which the operator is required to tele-operate the robot to pick the goals from a table. Positions of the target objects on the table are assumed to be known, as they can be computed using computer vision techniques such as [19].

Multiple demonstrations corresponding to picking of 4 different objects from the table via teleoperation were captured. These demonstration trajectories were further used to learn the neural network policy using the technique described above. Once the policy is learnt successfully, it is used in the teleoperation setting to assist the operator in performing the tasks (Architecture of the same is shown if Fig. 1). Below we present the results corresponding to AL-iLQR and Neural network for picking task.

B. AL-iLQR and learned policy for tracking operator reference

Proposed optimal control formulation for shared control solved using MPC methodology and the associated learnt policy using the Hamiltonian formulation were tested against multiple operators commanded trajectories. The plot in Fig. 2, shows the 2D plot of XY trajectories corresponding to one of the sampled operator's commanded trajectory. The trajectories corresponding to learned policy and the AL-iLQR approach were generated by simulating the robot motion. It can be seen that both the trajectories closely follow the operator's commanded trajectory within close proximity and smoothly transition to the goal as operator reaches the goal position, thus successfully assisting the operator in picking the object while not requiring the operator to finely manipulate the robot to reach the goal. Additionally, it can also be noticed that the trajectory corresponding to the learned policy is more compliant with the operator's trajectory than the one generated by the AL-iLQR algorithm, this behaviour is mainly due to learned policy being trained using multiple trajectories from collected demonstration and augmentation steps.

C. Reference tracking using learned policy

Figure 3 presents a 3D trajectory followed by the robot using learned policy against the trajectory commanded by the operator, and the corresponding plot of variation of x-, y- and z-coordinates against time are shown in Fig. 4. It can be seen from these plots that the robot closely tracks the commanded operator's inputs, and at the near terminal phase, robot is guided smoothly to the accurate location of the goal even though the operator's commanded position is offset from the goals location.

D. Robustness of the learned policy to noise and different starting position

The policy was also tested against varying operator trajectories with induced noise to test it's robustness against noisy operators inputs. Additionally it was also tested on the trajectories that were never part of the dataset used for training. The plot in Fig. 5 shows the 2D xy-plot of the noise induced multiple operator's scaled and rotated trajectories and the corresponding learnt policy generated trajectories and the variation of distance between these trajectories with respect to the goal is shown in Fig. 6. The distance variation

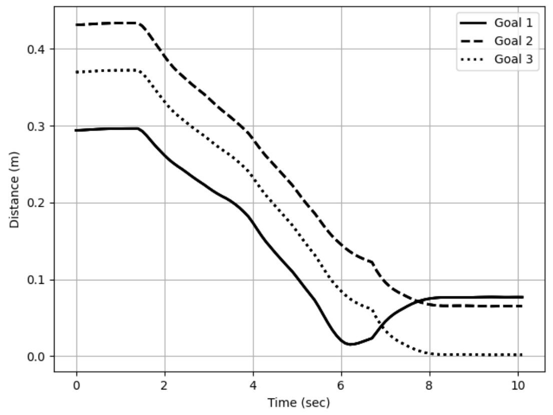

Fig. 9: Variation of distance between the robot and closely located goals

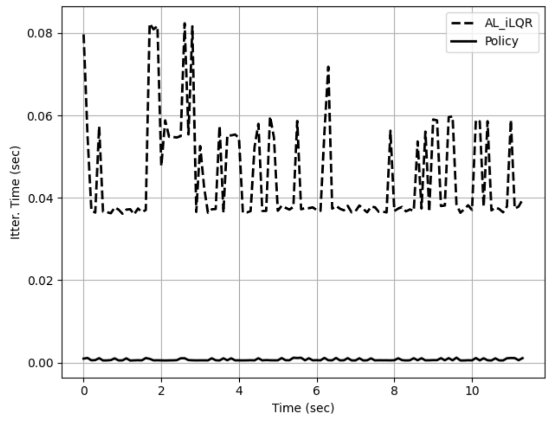

Fig. 10: Time consumed in every iteration by the AL-iLQR algorithm and the learned neural network policy

plot shows that even in the presence of induced noise, the learnt policy is able to drive the robot to the goal position. Further, additional validation of the learned policy was done by testing on a novel trajectory generated by randomly adding a bias and scaling the trajectory from the training dataset by a factor of 1.2. Fig. 7 shows the ability of the learnt policy to drive the robot on one of these completely novel trajectory. Thus, based on these results we can confidently conclude that the proposed approach allows us to learn an assistive policy to teleoperate the robot while simultaneously assisting in reaching the goal.

E. Closely packed goals scenario

Since the policy was trained using only single goal at a time, we tested the policy by placing closely packed multiple goals and picking one of the goal. Fig. 8 shows the plot of xy-trajectories corresponding to one of the scenario considered. It can be seen that, as the operator approaches the intended goal, the robots trajectory diverged towards the *Goal 1*, but due to continued motion of the operator, the policy never converges with this non indented goal, but converges with the goal *Goal 3* which is closest to operator when the operator stops moving. This behaviour can also be seen clearly through the plot in Fig. 9 showing the variation of distance between the robot and the goal positions. It can be seen that the distance between the *Goal 1* and the robot approaches to zero, but due to continued motion of the operator to approach *Goal 3*, the robot is guided to move to *Goal 3* which is confirmed by zeroing of the corresponding distance. This behaviour of the robot is influenced by the designed cost function, even though initially the robot is driven closer to the *Goal 1* which causes the weight function defined by eq. (2) to transition to goal tracking, the operator's control tracking term ensures that the robot continues to track opeator's commanded motion thus ensuring that robot does not converge to the unintended goal.

F. Comparison of computation time

Plot in Fig. 10 shows the plot of the time consumed for each iteration of the two algorithms. Both the algorithms were run on a standard laptop with intel i7 processor and 32GB RAM, running Ubuntu 22.04 operating system and both the algorithms were programmed using python 3.10. It can be seen from the plot that AL-iLQR algorithm takes on an average approximately 43 ms per iteration, while the neural network policy takes only 0.6ms, thus proving the computational advantage.

V. Conclusion

In this paper we introduced a novel approach for shared autonomy that utilises an optimal control formulation for adaptation. We train a shallow neural network based policy utilising the Hamiltonian of proposed optimal control problem as a loss function and few demonstration trajectories. The nature of the problem formulation and policy learning approach followed makes it robust enough to complete the assistive control tasks. This was validated by running a series of experiments on various scenarios. Although the results are preliminary in nature it shows promising results for future scenarios which includes it's application to continuous goal operations such as liquid pouring, door opening etc. Besides we also plan on performing user studies in order to better validate our approach on qualitative and quantitative measures such as operator satisfaction, task completion time, distance travelled. We also plan on extending this setup towards dual arm goal coordinated and bi-manual and goal coordinated operations.

References

[1] P. Desbats, F. Geffard, G. Piolain, and A. Coudray, "Force-feedback teleoperation of an industrial robot in a nuclear spent fuel reprocessing plant," *Industrial Robot: An International Journal*, 2006.

[2] T. Imaida, Y. Yokokohji, T. Doi, M. Oda, and T. Yoshikawa, "Ground-space bilateral teleoperation of ets-vii robot arm by direct bilateral coupling under 7-s time delay condition," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 499–511, 2004.

[3] C. Domingues, M. Essabbah, N. Cheaib, S. Otmane, and A. Dinis, "Human-robot-interfaces based on mixed reality for underwater robot teleoperation," *IFAC Proceedings Volumes*, vol. 45, no. 27, pp. 212–215, 2012.

[4] M. C. Çavuşoğlu, W. Williams, F. Tendick, and S. S. Sastry, "Robotics for telesurgery: second generation berkeley/ucsf laparoscopic telesurgical workstation and looking towards the future applications," *Industrial Robot: An International Journal*, vol. 30, no. 1, pp. 22–29, 2003.

[5] M. Selvaggio, M. Cognetti, S. Nikolaidis, S. Ivaldi, and B. Siciliano, "Autonomy in physical human-robot interaction: A brief survey," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7989–7996, 2021.

[6] A. D. Dragan and S. S. Srinivasa, "A policy-blending formalism for shared control," *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 790–805, 2013.

[7] S. Javdani, H. Admoni, S. Pellegrinelli, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization for teleoperation and teaming," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 717–742, 2018.

[8] S. Reddy, A. D. Dragan, and S. Levine, "Shared autonomy via deep reinforcement learning," *arXiv preprint arXiv:1802.01744*, 2018.

[9] Y. Du, S. Tiomkin, E. Kiciman, D. Polani, P. Abbeel, and A. Dragan, "Ave: Assistance via empowerment," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4560–4571, 2020.

[10] W. Tan, D. Koleczek, S. Pradhan, N. Perello, V. Chettiar, V. Rohra, A. Rajaram, S. Srinivasan, H. S. Hossain, and Y. Chandak, "On optimizing interventions in shared autonomy," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 5341–5349, 2022.

[11] C. Schaff and M. R. Walter, "Residual policy learning for shared autonomy," *arXiv preprint arXiv:2004.05097*, 2020.

[12] S. Chen, J. Gao, S. Reddy, G. Berseth, A. D. Dragan, and S. Levine, "Asha: Assistive teleoperation via human-in-the-loop reinforcement learning," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 7505–7512, IEEE, 2022.

[13] P. M. Kebria, S. Al-wais, H. Abdi, and S. Nahavandi, "Kinematic and dynamic modelling of ur5 manipulator," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 004229–004234, 2016.

[14] M. Rubagotti, T. Taunyazov, B. Omarali, and A. Shintemirov, "Semi-autonomous robot teleoperation with obstacle avoidance via model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2746–2753, 2019.

[15] R. Lima, V. Vakharia, U. Rai, H. Mehta, V. Vatsal, and K. Das, "Model-mediated delay compensation with goal prediction for robot teleoperation over internet," in *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1037–1043, IEEE, 2023.

[16] D. Bertsekas, *Dynamic programming and optimal control: Volume I*, vol. 4. Athena scientific, 2012.

[17] J. Carius, F. Farshidian, and M. Hutter, "Mpc-net: A first principles guided policy search," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2897–2904, 2020.

[18] T. A. Howell, B. E. Jackson, and Z. Manchester, "Altro: A fast solver for constrained trajectory optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7674–7679, IEEE, 2019.

[19] S. S. Rambhatla, I. Misra, R. Chellappa, and A. Shrivastava, "Most: Multiple object localization with self-supervised transformers for object discovery," 2023.