2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Local Linearity is All You Need (in Data-Driven Teleoperation)

Michael Przystupa[1,2], Gauthier Gidel[3], Matthew E. Taylor[2,5],
Martin Jagersand[2], Justus Piater[4], Samuele Tosatto[4]

***Abstract*— One of the critical aspects of assistive robotics is to provide a control system of a high-dimensional robot from a low-dimensional user input (i.e. a 2D joystick). Data-driven teleoperation seeks to provide an intuitive user interface called an *action map* to map the low dimensional input to robot velocities from human demonstrations. Action maps are machine learning models trained on robotic demonstration data to map user input directly to desired movements as opposed to aspects of robot pose ("move to cup or pour content" vs. "move along x- or y-axis"). Many works have investigated nonlinear action maps with multi-layer perceptrons, but recent work suggests that local-linear neural approximations provide better control of the system. However, local linear models assume actions exist on a linear subspace and may not capture nuanced motions in training data. In this work, we hypothesize that local-linear neural networks are effective because they make the action map *odd* w.r.t. the user input, enhancing the intuitiveness of the controller. Based on this assumption, we propose two nonlinear means of encoding odd behavior that do not constrain the action map to a local linear function. However, our analysis reveals that these models effectively behave like local linear models for relevant mappings between user joysticks and robot movements. We support this claim in simulation, and show on a realworld use case that there is no statistical benefit of using non-linear maps, according to the users experience. These negative results suggest that further investigation into model architectures beyond local linear models may offer diminishing returns for improving user experience in data-driven teleoperation systems.**

I. Introduction

The choice of the controllable coordinate system in teleoperation can dramatically affect the difficulty of robotic manipulation problems. For example, wiping a table is often easier to control directly in the corresponding 2D Cartesian plane with respect to a table, as opposed to the native N-dimensional joint space of the robotic arm. This observation motivates typical *mode switching* mappings, in which users control subsets of the robot pose at any time [1]. These mappings are the *de facto* scheme in assistive robotic systems and are often intuitive as user inputs map to the coordinate system of the robot's end effector. Such interfaces are valuable for users with limited mobility, whose teleoperation interfaces often have restricted available degrees of freedom.

Unfortunately, mode switching algorithms can be mentally taxing and challenging to scale. Previous results controlling a single manipulator suggest that mode switching can involve up to 30 – 60 mode switches to complete everyday tasks like opening doors [2], [3]. Transferring mode switching to settings such as when operating parallel serial manipulators would lead to even greater complexity. One solution would be increasing the user's number of available control inputs, but sometimes this is not feasible due to user physical limitations [4]. The limitations of mode switching have motivated research on data-driven *action mapping* algorithms that learn to project low-dimensional actions (e.g., input from a joystick) onto high-dimensional control commands (e.g., robotic joint velocities) [5], [6]. These models are typically learned with demonstration data of the desired task. Figure 1 provides a visualization of action maps during deployment.

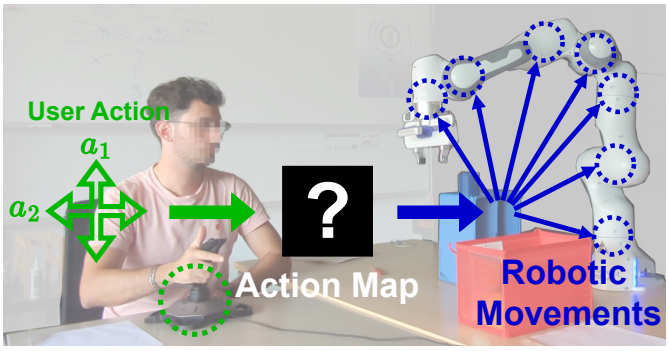

Fig. 1: User teleoperating a 7 joint robot from a 2 DoF joystick with a data-driven spatial mapping interface.

Early research focused on linear action maps to reconstruct the relevant action space. These methods are motivated by results modeling hand grasps, where one can approximate 80% of hand poses with only the first two loading vectors found via principal component analysis [7]. Previous research has applied these learned linear action maps to develop robotic control algorithms [8]–[11].

Recent work has proposed *state conditioned* action maps which adapt high-dimension commands given task-context information [5], [6], [12]–[15]. A conditional autoencoder (CAE) learns to reconstruct high-dimensional manipulations from states and a latent variable that maps the user's input. Several aspects have been studied, including user experience [16], [17], efficient data collection [18], and complex data conditioning (e.g., images or text) [13], [14]. CAE-based action maps have also been beneficial beyond assistive robotics in reinforcement learning (RL). Previous research shows action maps can improve sample efficiency [15], [19], bound action spaces to those observed in offline demonstration data [20], and autonomously learn hand synergies [21].

A limitation of the CAE approach is that demonstration data quality impacts the learned action map [6], but pre-

[1] przystupa@ualberta.ca, [2] University of Alberta, [3] Universite de Montreal, [4] University of Innsbruck, [5]Alberta Machine Intelligence Institute (Amii)

vious work finds that enforcing human teleoperation priors can mitigate the need for large, high-quality demonstration datasets [6], [12], [17]. Li et al. [17] exploit human priors to propose regularizer terms to help learn a remapping model between a previously trained action map and the user's preferences. Their results suggest such regularization terms improve the generalization of the action map with a limited amount of demonstration data. Przystupa et al. [12] investigated enforcing *teleoperation reversibility*, where users expect symmetrically opposite user input to undo prior robotic motions. The authors find that local linear action maps, i.e., functions linear w.r.t. user inputs and nonlinear to the robot's state, are better architectural choices.

In this paper, we are interested in furthering these previous investigations on neural network learning and structure to enforce human teleoperation priors. We propose both regularization and architecture modifications to have nonlinear action maps that enforce the *odd* functional behavior w.r.t. user inputs (e.g., $f(-a) = -f(a)$), like previously proposed local linear action maps [12]. In this paper, we test whether nonlinear odd action maps improve the user experience w.r.t. linear ones. Our analysis shows that the added nonlinearity in the action space does not provide significant benefit for the user. Not only that, but both our proposed approaches behave similarly to local linear models after training.

These findings are valuable for the development of deployable data-driven teleoperation systems. Local linear models are simpler to analyze, so if they are sufficient for teleoperation, this paves the way for accessibility of existing system analysis tools [22]. Applying such tools can guarantee properties previously proposed as necessary in data-driven teleoperation systems – such as controllability, scalability, and user input reversibility [6], [12], [17] – but have only been loosely empirically demonstrated as being enforced [5], [6]. As data-driven teleoperation involves human users, concrete guarantees of how the learning system will behave are necessary for any ethical approval of real-world applications.

II. Background

Previous works on data-driven teleoperation frame the problem as a Markov decision process where user inputs occur at discrete time steps [5]. We consider this an unnecessarily general framework for data-driven teleoperation. Our action maps technically operate in continuous time, and we do not meaningful study or estimate means of maximizing rewards of users in this paper. We describe the dynamical system induced by the learned action map because our commands operate on robot velocities. We then describe teleoperation properties and close with discussion of the deep learning conditional action map framework.

A. Teleoperation as Dynamical Control System

We can frame teleoperation in terms of a first-order ordinary differential equation,

$$\dot{x}(t) = f_\theta(x(t), a(t)),$$

where $x(t) \in \mathbf{R}^d$ represents the state of the robot (i.e., joint configurations or end-effector's pose), $\dot{x}(t) \in \mathbf{R}^d$ is velocity, $a(t) \in \mathbf{R}^n$ is the user action (e.g., the 2D position of the joystick as in Fig. 1), $t \in \mathbf{R}^+$ is the time, and the action-mapping $f_\theta : X \times A \to \dot{X}$ is the learned mapping from states X and user actions A to robot velocities $\dot{X}$.

We assume that for teleoperation, the dynamical system evolves following Euler's method, i.e.,

$$x(t + \tau_i) = x(t) + \tau_i f_\theta(x(t), a(t)),$$

where $\tau_i \in \mathbf{R}^+$ is the discretization step. We consider these modeling assumptions reasonable, as the teleoperation context is a closed-loop system, where the trajectory is determined online by the internal policy $\pi(o_h(t)) = a(t)$ of an unknown human observation $o_h(t) \in \mathbf{R}^m$.

B. Desirable Teleoperation Properties

Given the dynamical system described above, we provide definitions of previously suggested teleoperation properties [5], [6], [12], [17]. These properties provide mathematical definitions to help guide the design of data-driven teleoperation systems. Excluding controllability, previous works have guaranteed these properties by modifying the training algorithm or neural architecture [12], [17].

Controllability: Users expect to be able to reach arbitrary robot configurations, or that from any x exists a sequence $A = \{a(0), a(t_1), \ldots, a(t_{T-1}), a(t_T)\}$ in the action mapping $f_\theta(x(t), a(t))$ to reach desired target $x^\star$.

Monotonicity of User Inputs Scaling: As the user's action increases, the action map output magnitude increases, and as the user's action decreases, the magnitude of the action map decreases, i.e., for fixed x and a with $\alpha_1, \alpha_2 \in \mathbf{R}$, if $|\alpha_1| < |\alpha_2|$ then we have $\|f(x, \alpha_1 a)\| \leq \|f(x, \alpha_2 a)\|$.

Consistency: The dynamical system velocities should change smoothly w.r.t to the state for fixed user inputs. This is achievable if we can assume f is Lipschtiz continuous ($||f(x, a) - f(x', a)|| \leq L\|x - x'\|$ for $L \in \mathbf{R}^+$). Bounding the Lipschitz constant is achievable during training with the algorithm of Gouk et al. [23], which we apply in our experiments like previous work [12].

(User Input) Reversibility: Users are able to *undo* their inputs; for state $x(t)$ and $x(t + \tau_i) = x(t) + \tau_i f(x(t), a(t))$, if $a(t + \tau_i) = -a(t)$ then we expect $x(t + \tau_i + \tau_{i+1}) = x(t)$. The discretization of the continuous-time system makes reversibility hard to achieve. A soft (looser) version of this property has been previously proposed [12]. Enforcing this property is achievable with odd functions of user actions and is a motivating reason for our interest in developing odd nonlinear action maps.

C. Deep Conditional Action Maps

Deep conditional action maps use neural networks to learn the action map model. The notation for variables dependent on time is removed because it is irrelevant for learning action maps. We assume access to a dataset $D = \{(x_i, \dot{x}_i)\}_{i=1}^N$ where $x_i = x(\tau_i), \dot{x}_i = \dot{x}(\tau_i)$ are pairs of robot states and velocities collected at time τ_i through kinesthetic teaching for a desired task. The lack of user actions in the dataset makes the problem of finding an optimal action map unsupervised.

The action map f_θ is learned as a conditional autoencoder (CAE), where the user action a, which is not in the training data, is treated as a latent variable. The conditional autoencoder is composed of an encoder $g_\theta : X \times \dot{X} \rightarrow A$, which generates latent actions based on the current state and velocity, and a decoder $f_\theta : X \times A \rightarrow \dot{X}$, which corresponds to the action map. The autoencoder minimizes the loss:

$$L(x, \dot{x}, f, g) = L^{recon}(\dot{x}, f(x, g(x, \dot{x}))) + L^{reg}(f, g, x, \dot{x}),$$

where $L^{recon} = \frac{1}{2}||\dot{x} - f(x, g(x, \dot{x}))||_2^2$ is the reconstruction loss, and typically the mean-square error is sufficient. Previous work has considered different regularizers L^{reg}, including the Kullback-Liebler divergence or latent dynamics models [6], [15]. In this paper, we propose to use a regularizer that encodes desirable teleoperation properties, which improve the user experience.

An important distinction is the notion of end-to-end versus hypernetwork decoder. End-to-end decoders concatenate both x and a and input them into a multilayer perceptron. A hypernetwork decoder [24] decomposes action and context interactions: given the robotic state x, a local function is predicted to reconstruct actions. In previous work, this local function was linear: $f_\theta(x, a) = \phi_\theta(x)a$, where $\phi_\theta(x) : \mathbf{R}^d \rightarrow \mathbf{R}^{d \times n}$ [12]. We refer to this as a *hyper-linear* model when the hypernetwork predicts just a linear function.

III. Building Odd Nonlinear Action Maps

This section describes two ways to learn action maps that lead to odd functional behavior that do not assume a linear function. The first approach adds regularizers to an end-to-end decoder to encourage the action map to be odd in user inputs and maps zero input to zero movements. This approaches is more common in deep learning, where action map properties are enforced through the loss function. The second method generalizes the hyper-linear action map through additional assumptions to allow stacking layers.

A. Odd Function Regularization

A natural means to enforce teleoperation properties in a neural network is the specification of the loss function. We propose the following regularizer $L^{reg}(f, g, x, \dot{x}) =$

$$||(-\dot{x}) - f(x, -a)||_2^2 + ||f(x, g(x, 0))||_2^2 + ||g(x, 0)||_2^2.$$

Our proposed regularizer achieves two behaviors: (1) enforcing odd functional behavior to enable reversing and (2) encouraging that zero inputs map to zero outputs. We include this regularizer in each mini-batch by adding $\dot{x} = 0$ and inverse tuples $(-\dot{x}, -a)$ with the state inputs in the batch.

Unlike other general regularizer terms, our regularizer explicitly encourages desired teleoperation properties. The model learns to behave as an odd function, which is crucial for achieving a form of user input reversibility. Other reported regularizer terms only compress the action space [6] or constrain a black-box dynamics model [15], but otherwise never explicitly encourage any teleoperation properties.

However, other prior teleoperation regularizers have been proposed in the work of Li et al. [17]. We do not empirically compare them in this paper for a number of reasons. Their regularizer assumes kinematics knowledge to define which increases complexity in the training procedure. They consider the task space, which is ambiguous under kinematics and may not have one-to-one relations between the robot velocity commands and task space movement. In contrast, our regularizer does not require kinematics and only uses values produced by the CAE. We regularize in the robot joint space where our user's inputs are mapped, which leads to desired behaviors in task space. Previous work has also suggested that the kinematics regularizers of Li et al. [17] offer minimal benefit when training action maps [12] as opposed to fine-tuning an existing action map to user preferences which the work of Li et al. [17] investigated.

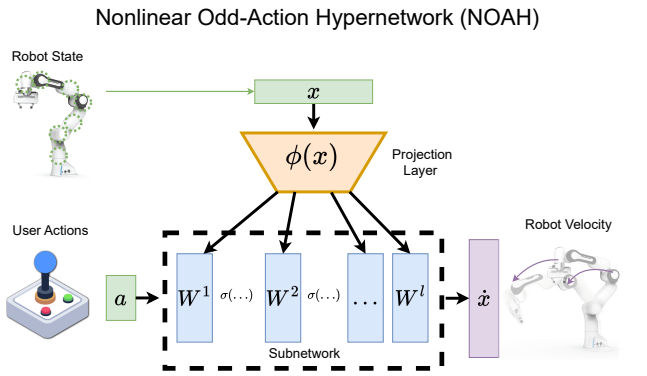

Fig. 2: NOAH ensures odd actions: the projection module generates the subnetwork parameters, which is composed of a sequence of layers that preserve the odd property by using odd activations. Note that, during training, user actions are generated with an encoder.

B. Building Odd Nonlinear Subnetworks

An alternative to changing the loss function is by using hypernetworks to predict subnetworks thats enforce desired properties. The hypernetwork predicts the weights of the subnetwork based on the robot state. This subnetwork is then composed of n layers, and only takes in user inputs.

Each layer in the subnetwork consists of a hyper-linear set of weights, which can be formed by reshaping the hypernetwork prediction $\phi_\theta(x) = w^{DN}$ into a matrix, but in this paper we treat the operation as a 3D tensor product,

$$W^i(x) = H^i \otimes \phi_\theta(x) + B^i,$$

where $W(x) \in \mathbf{R}^{h \times n}$ is the resulting subnetwork layer weights after tensor product $\otimes$ between $H \in \mathbf{R}^{h \times w \times n}$ and hypernetwork features $\phi_\theta(x) \in \mathbf{R}^w$. We learn a matrix bias $B^i \in \mathbf{R}^{h \times n}$ in the hypernetwork, but these are different from predicting a bias term in the subnetwork weights. Bias terms in the subnetwork would violate our constraint to have an odd function of user inputs.

In addition to the hyper-linear weights, all activation functions σ of the subnetwork must be odd in order for the action map to be odd end-to-end. This means activation functions that are asymmetrical like rectified linear units or

sigmoid cannot be used. We must then choose activation functions like tanh, sinusoidal functions or odd polynomials.

Combining these two observations, we can construct the action map as an arbitrary depth subnetwork as follows:

$$\dot{x} = W^n(x) \circ h^{n-1} \circ h^{n-2} \circ ...h^2 \circ h^1,$$

where $h^i(a) = \sigma(W^i(x)a)$ is the i-th hidden layer of the subnetwork. We visualize this architecture in Figure 2. For clarity, in our experiments we use a three-layer model: $\dot{x} = W^3(x)\sigma(W^2(x)\sigma(W^1(x)a))$, and share state features $\phi_\theta(x)$ between each h^i for parameter efficiency. We refer to these conditional action maps as *nonlinear odd action hypernetwork* (NOAH) in our experiments.

C. Are Nonlinear Models of User Actions Necessary?

Although we propose two means of imbuing odd functionality into conditional action maps, a pertinent question is whether users would substantially benefit from using these interfaces. We hypothesize that *there will be no statistically significant difference between our proposed nonlinear action map methods and local linear models in user experience*. One reason for this hypothesis is that the hyper-linear decoders are known to achieve previously listed teleoperation properties: monotonicity, consistence, and (soft) reversibility, giving users significant control over the system. It is also plausible that during optimization the subnetworks converge to behave effectively as a local-linear function which are in the set of odd functions. This is easily seen by looking at the first order Taylor series approximation at $a_0 = 0$:

$$\begin{aligned} f(x,a) &\approx f(x,a_0) + J^a(x,a_0)(a-a_0) \\ &= f(x,0) + J^a(x,0)a, \end{aligned}$$

where $f(x,0) = 0$ by being an odd function. We investigate this in our experiments to test how well the local-linear approximation compares to the learned nonlinear model.

Furthermore, we show that as the discretization step decreases, the induced robotics' trajectory under the linearization and original model become the same. We show this by deriving the error between the final state of the linearization and the original model when following the same user action sequence. Proofs are included in the appendix. The result relies on the following lemma to bound the error between the predictions of linearization at $\hat{x}(t)$ and true model at $x(t)$:

Lemma 1. *For two states $x(t)$ and $\hat{x}(t)$, the action $a(t)$ under a Lipschitz continuous, odd, action map f and linearization $\hat{f} = \nabla_a f(\hat{x}(t),a)|_{a=0}a(t)$ have $\|f(x(t),a(t)) - \hat{f}(\hat{x}(t),a(t)\| \leq M+LC$, where $M = \max_a \|f(x(t),a(t))\|$, L is the Lipschitz constant, and $C = \max\{\|a\| : a \in A\}$.*

The intuition of this lemma is that in a robot's state space for a Lipschitz action map with bounded user inputs, there exists some maximal distance of robot velocities between two arbitrary robot states. With this result, we can then prove that as the discretization step $\tau \to 0$ these errors disappear.

Theorem 1. *For an odd action map f that has Lipschitz constant L on a bounded action space, following Euler integration, the distance between trajectory $x(t)$ and $\hat{x}(t)$ where $\hat{X}$ follows linearization $\hat{f}$ off, we have $\|x(t)-\hat{x}(t)\| \leq \tau(T-1)(M+LC)$, where τ is the discretization step and T is the number of discrete steps taken.*

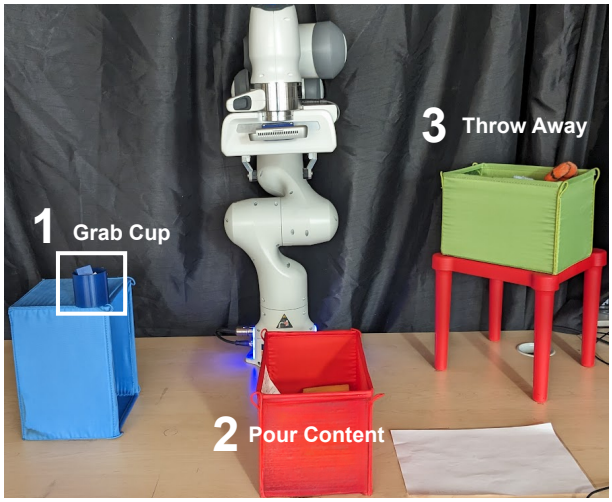

Fig. 3: The user study task consists of three phases: (1) grab the cup, (2) pour its content in the red bin, and (3) dispose of the cup in the green box.

This theorem suggests, particularly for shorter trajectories, users likely would not notice substantial changes in motion between the model and it's local linear approximation.

IV. Experiments

In this section, we report the results of experiments evaluating the proposed nonlinear action mapping approaches. We compare against previously proposed state-conditioned linear (SCL) maps as a baseline [12]. SCL is a local linear map that transforms the matrix predictions with the Gram-Schmidt process at deployment. We refer to the MLP introduced in Section II-C as *AE* or hypernetwork models introduced in Section III-B as *NOAH*, and include (*Loss*) for models trained with the proposed regularizers (Section III-A).

We first evaluated the quality of action maps with empirical metrics in simulation. We measured each model's prediction accuracy and the effect of the local linear approximation away from $a_0 = 0$. We then conducted simulated teleoperation experiments to investigate how well the models could track trajectories. We close with results from our user study that evaluated our hypothesis on non-linearity's benefit with able-body participants, testing whether statistical significance exists in subjective and objective metrics.

In all experiments, we used the same hyperparameters to train each system. Each model had three linear layers with tanh activations. NOAH models had 32 units in the subnetwork layers and 48 in the hypernetwork layer, whereas SCL and AE models had 256 neurons per layer. These layer parameters keep the total parameters similar between models, but NOAH models had fewer parameters (approximately 65,000 vs. 70,000 for other models). We trained each model for 1000 epochs for regression and simulated teleoperation

	Bottle to Shelf	Open Box	Plates Standing to Lying	Pouring	Scoop
SCL	$\mathbf{-5.727 \pm 0.010}$	-5.981 ± 0.020	-5.451 ± 0.010	-5.395 ± 0.005	-5.832 ± 0.015
AE	-5.546 ± 0.007	$-5.761 \pm 0.002^*$	-5.458 ± 0.002	$-5.303 \pm 0.027^*$	-5.702 ± 0.005
AE (Loss)	$-5.420 \pm 0.171^*$	$-5.264 \pm 0.219^*$	$-5.312 \pm 0.132^*$	$-5.149 \pm 0.107^*$	$-5.263 \pm 0.148^*$
NOAH	-5.609 ± 0.098	-5.996 ± 0.012	$\mathbf{-5.500 \pm 0.089}$	$\mathbf{-5.480 \pm 0.010}$	-5.717 ± 0.117
NOAH (Loss)	-5.720 ± 0.006	$\mathbf{-5.998 \pm 0.024}$	-5.436 ± 0.011	-5.421 ± 0.009	$\mathbf{-5.843 \pm 0.005}$

TABLE I: Log Base 10 Mean Square Test Error reconstructing robot velocities (lower is better). Bold text are models with lowest mean and asterisk (*) indicate statistical signifance compared to other models by the Tukey's range test ($p < 0.05$).

experiments, and 500 epochs for the user study experiments. Models are trained with mini-batches of 256 with the Adam optimizer with a learning rate of 0.001.

A. Action Map Analysis

This subsection reports results analyzing the quality of our proposed action mapping algorithms performed in simulation. A limitation of these results is that they could be unrealistic because we can predict optimal actions under the model via the trained encoder or apply optimization techniques that may not accurately reflect human behaviors (e.g., simulating several trajectories under the action map).

However, the goal of our simulation results are to understand each action map under reproducible conditions. Our regression experiments show empirically how well we can reconstruct a desired velocity supposing we knew the optimal action, which is necessary to effectively complete the desired task. Our simulated teleoperation experiments help determine the generalization of our action maps. Then the target test states are within the demonstration data distribution, so it is valuable to determine at least for a subset of states whether our action map is controllable (see Section II-B).

1) Model Regression Quality: Our first interest was to analyze the predictive quality of our model. We reconstructed publicly available demonstration trajectories from the work of Sayantan et al. [25]. Each dataset consists of 10 trajectories with 1000 samples each (10000 interaction tuples). Robot commands are the finite differences between every third sample ($\dot{x}_i \approx x_{i+3} - x_i$) in collected demonstrated trajectory. We found this easier to control in preliminary user study experiments compared to sensor velocities. For each demonstration in the dataset, we use 90% of interaction tuples for training, 5% for validation, and 5% for testing. We report the test mean square error in Table I where we also report statistical significant results ($p < 0.05$) by the Tukey's range test against the action map with lowest mean performance. We find NOAH generally had the lowest mean performances, but these results typically were only significant compared to AE models.

We further compare the effect of the linearization of each action map on the output prediction to determine if learned subnetwork function is effectively behaving linearly. We compare the linear approximation against the model prediction on the test dataset with cosine-sim$(f, x, a) =$

$$\frac{f(x,a) \cdot (f(x,0) + (\nabla_a f(x,a)|_{a=0})a))}{\|f(x,a)\|\|f(x,0) + (\nabla_a f(x,a)|_{a=0})a\|}.$$

We increase the magnitude of $\|a\|$ at each testing state to see how cosine similarity varies in Figure 4. We included results on action maps trained with data from the user study in this analysis of linearization because it was relevant to justifying our hypothesis with human users.

We concluded that the underlying direction of robot commands is largely unchanged under the approximation, except at much larger actions (e.g., NOAH (Loss) or AE (Loss)). This suggested that from a user's perspective, the teleoperation experience under the local linear approximation would be similar in terms of the generated robot movements.

2) Simulated Teleoperation: In our last set of simulation experiments, we ran an artificial teleoperation experiment to use the action map to reach a specified target $x^\star$ in the test datasets. User inputs were selected by choosing the best input between a set A of n standard distributed actions, i.e.,

$$a_t = \arg\min_{a \in A} ||x^\star - x_t - \nu f(x_t, a)||_2^2.$$

We found this could be unstable in practice due to local minima, and mitigate this effect with via-points from the test trajectory to improve stability of simulated teleoperation. The mean performances are shown in Figure 5 in the demonstration datasets with their optimal action norm size. Figure 5 also shows the effect of the size of the action norms for both AE (Loss) and NOAH in reaching the target states compared to the linearization approximation.

B. User Study Results

To validate our hypothesis on the impact of action map structure, we conducted a user study where participants completed a sequential multi-stage task using an *Emika Franka Robot* [26].[1] Figure 3 shows the experimental setup. We asked participants to pick up a cup from the top of a box (blue), empty its contents in the middle container (red), and throw the cup away in the corner on the opposite side of the testing space (green). This task transitioned between multiple aspects of the Cartesian pose for each subtask. We collected 32 trajectories for a total of 9270 interaction tuples, or approximately 289 steps per trajectory to train the data-driven teleoperation mappings.

We had 15 participants (13 male, two female) ages 20 - 55. Each participant used four different teleoperation mappings during the experiment: mode switching, NOAH, AE (Loss), and SCL. Participants completed four trials with

[1]The studies were approved by University of Alberta Research Ethics and Managements (Pro00054665).

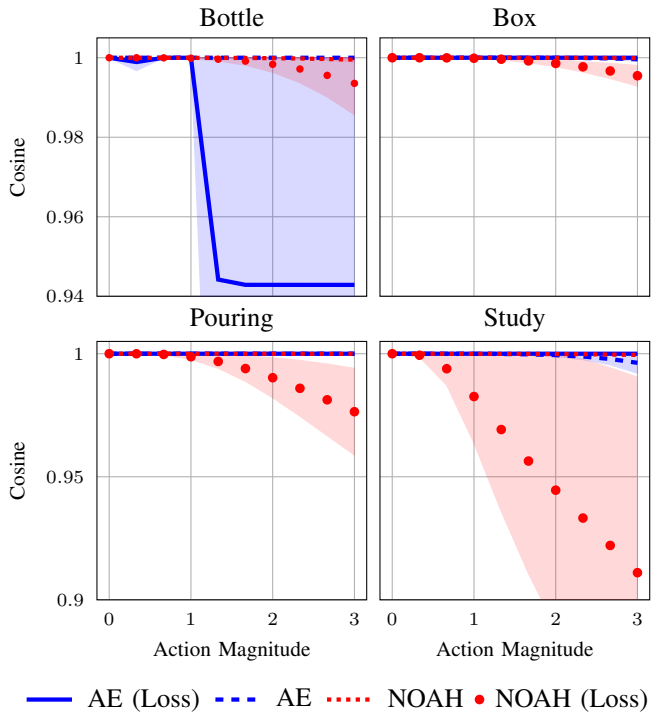

Fig. 4: Cosine similarity between prediction and linearized approximation averaged over 5 random seeds. Exhibiting high cosine similarity suggests the predictions generate similar movements for the same user inputs.

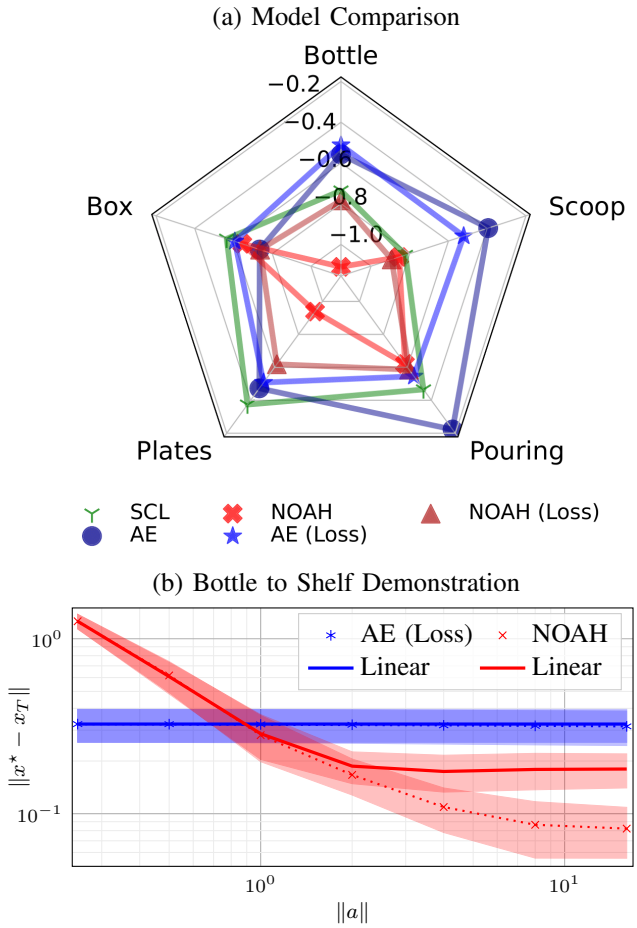

Fig. 5: Simulated teleoperation results measuring distance to target robot pose $x^\star$. (a) Shows final distance ($\log_{10}$ scale) for each model with best tuned user-action norm (b) Compares Nonlinear maps performances against their linearized approximations.

each mapping function, where the first two trials were two minutes (training trials) and the latter two were three minutes (testing trials). Mode switching consisted of six modes: three for translation and three for orientation. In our evaluation process, every participant first completed the mode switching trials, and afterwards performed trials with the three data driven mappings in a randomized single-blind set-up. We chose to have mode switching first to help familiarize users with the joystick interface and establish baseline expectations across users to fairly evaluate the data-driven mappings. We included an option to reset the robot to the initial pose at any time during the trial, which meant participants could reset between subtasks instead of performing the movement end-to-end. This strategy was not necessarily optimal because no demonstrations were collected to perform the second and third subtasks from the start pose.

We collected several subjective metrics, including the NASA-TLX score and a subjective experience survey on a 7-point Likert scale. The keywords in our Likert questions were: intuitive, precise, smoothness, undo, and control. Figure 6 shows violin-plots of participants' NASA-TLX scores for each action map. We found no statistical significance between the four action maps by the Dunn median statistical test for a p-value of 0.05. We show the statistically significant Likert scale results: control between SCL and NOAH and smooth between mode switching and AE in Figure 7.

Furthermore, we measured success rates of the entire task and subtasks. Complete success with any system required successful completion of each of the three subtasks. We break down results into several scenarios based on our findings. The significant categories included full success or just missing pouring, which was the usual failure case. We counted partial successes where participants completed some portions of the subtasks successfully and otherwise reported failures as no subtask completed. We show the percentage of trials for reported scenarios in Figure 8.

1) Discussion: User study results suggest that, compared to the previously proposed SCL method and mode switching, users had similar experiences with both NOAH and AE approaches. We found no statistical differences in our survey results or the NASA-TLX score, suggesting users had similar experiences across mapping functions. If we considered missing the pour as tolerable, the AE method could be considered objectively superior to NOAH and SCL where users only struggled pouring with the AE model.

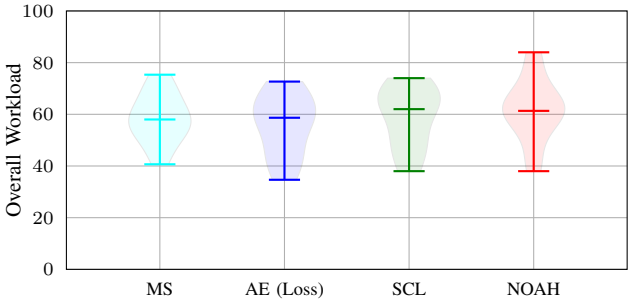

Fig. 6: NASA Workload Index between systems during user study. Between all system used by participants we did not find statistical significance with ($p > 0.05$) by the Posthoc Dunn mean test.

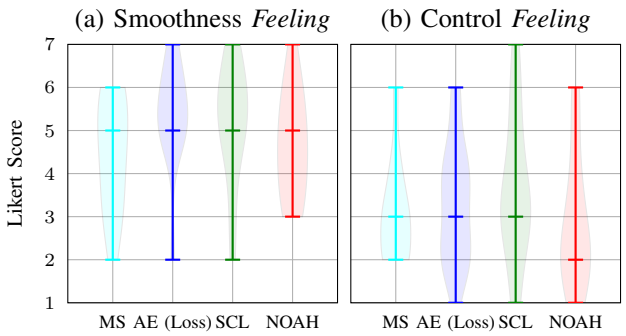

Fig. 7: Subjective Likert score results from user study. We found statistical significance ($p < 0.05$) for *control feeling* (a) between SCL and NOAH and *smoothness* (b) between MS and AE. All other pairs were insignificant. No statistical differences were found for other questions.

However, the data used in the experiment consists of three separately collected sets of similar movements that we combined in order for the AE (Loss) model to learn a deployable mapping. In contrast, both NOAH and SCL were generally able to learn better mappings with any of the three datasets we had combined. Each of these datasets only took around ten minutes to collect, but required several iterations to validate the mapping before experimental trials. This is an open challenge of our research as data-driven interfaces are sensitive to data quality and initialization. Model initialization could also affect the mapping between training runs, and investigating these affects in deployment is a promising future work.

V. Conclusion and Future Work

In this work, we proposed a novel neural architecture and regularizer terms that enforce odd functional behavior of nonlinear action maps. Our results both in simulation and from our user study suggest that although it is possible to create odd nonlinear action maps, their benefits are minimal. We found NOAH could sometimes be better at reaching targets in our simulated teleoperation experiments, but as a regressor model, prediction accuracy was not statistically

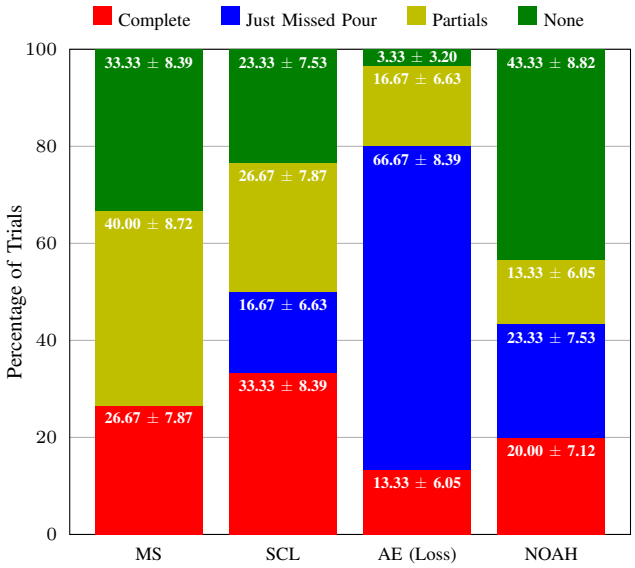

Fig. 8: Success rates for each task completion type for teleoperation mappings. Standard deviation is calculated as treating each setting as a Bernoulli variable.

better than previously proposed SCL across demonstration datasets. Our linearization analysis revealed that the direction of robotic movements was not impacted drastically between the nonlinear model and its local linearization. Our user study results further support our hypothesis because users did not show a strong preference between one action map to another. These findings help validate our simulation findings showing minimal gains with nonlinear odd action maps.

Results suggest local linear models are sufficient for data-driven teleoperation, but research into other aspects of data-driven teleoperation is necessary. To determine measurable architecture gains, more rigorous standards for data teleoperation mapping comparisons are a crucial next step. Identifying a means to automate correlating the quality of action maps to user experience is a promising future work direction because the cost to conduct user studies is nontrivial for testing every change to a model. As robots become more common for both able-bodied individuals and those living with a disability, it is increasingly important that lay people be able to drive multi-degree-of-freedom platforms with low-dimensional teleoperation controllers. Action maps are a promising direction for software solutions to learn teleoperation mapping functions, but much work is needed for these methods to be usable in the real world.

References

[1] Benjamin A. Newman, Reuben M. Aronson, Siddhartha S. Srinivasa, Kris Kitani, and Henny Admoni. Harmonic: A multimodal dataset of assistive human–robot collaboration. *The International Journal of Robotics Research*, 41(1):3–11, 2022.

[2] Laura V. Herlant, Rachel M. Holladay, and Siddhartha S. Srinivasa. Assistive teleoperation of robot arms via automatic time-optimal mode switching. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 35–42, 2016.

[3] Francois Routhier and Philippe S. Archambault. Usability of a joystick-controlled six degree-of-freedom robotic manipulator. In *RESNA ANNUAL Conference 2010*, 2010.
[4] Hiroki Higa, Kei Kurisu, and Hideyuki Uehara. A vision-based assistive robotic arm for people with severe disabilities. *Transactions on Machine Learning and Artificial Intelligence*, 2(4):12–23, 2014.
[5] Dylan P. Losey, Hong Jun Jeon, Mengxi Li, Krishnan Srinivasan, Ajay Mandlekar, Animesh Garg, Jeannette Bohg, and Dorsa Sadigh. Learning latent actions to control assistive robots. *CoRR*, abs/2107.02907, 2021.
[6] Dylan P. Losey, K. Srinivasan, Ajay Mandlekar, Animesh Garg, and D. Sadigh. Controlling Assistive Robots with Learned Latent Actions. *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020.
[7] Marco Santello, Martha Flanders, and John Soechting. Postural hand synergies for tool use. *The Journal of Neuroscience*, 18:10105–15, 12 1998.
[8] Panagiotis K. Artemiadis and Kostas J. Kyriakopoulos. Emg-based control of a robot arm using low-dimensional embeddings. *IEEE Transactions on Robotics*, 26(2):393–398, 2010.
[9] Matei Ciocarlie and Peter Allen. Hand posture subspaces for dexterous robotic grasping. *I. J. Robotic Res.*, 28:851–867, 06 2009.
[10] Aggeliki Odest and Odest Jenkins. 2d subspaces for user-driven robot grasping. *Robotics, Science and Systems Conference: Workshop on Robot Manipulation*, 2007.
[11] Giulia Matrone, Christian Cipriani, Maria Chiara Carrozza, and Giovanni Magenes. Real-time myoelectric control of a multi-fingered hand prosthesis using principal components analysis. *Journal of neuroengineering and rehabilitation*, 9:40, 06 2012.
[12] Michael Przystupa, Kerrick Johnstonbaugh, Zichen Zhang, Laura Petrich, Masood Dehghan, Faezeh Haghverd, and Martin Jagersand. Learning state conditioned linear mappings for low-dimensional control of robotic manipulators. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 857–863, 2023.
[13] Siddharth Karamcheti, Albert J. Zhai, Dylan P. Losey, and Dorsa Sadigh. Learning visually guided latent actions for assistive teleoperation. In *Proc. Conf Learning for Dynamics and Control*, pages 1230–1241, 2021.
[14] Siddharth Karamcheti, Megha Srivastava, Percy Liang, and Dorsa Sadigh. Lila: Language-informed latent actions. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1379–1390. PMLR, 08–11 Nov 2022.
[15] Arthur Allshire, Roberto Martín-Martín, Charles Lin, Shawn Manuel, Silvio Savarese, and Animesh Garg. LASER: learning a latent action space for efficient reinforcement learning. *CoRR*, abs/2103.15793, 2021.
[16] Hong Jun Jeon, Dylan Losey, and Dorsa Sadigh. Shared Autonomy with Learned Latent Actions. In *Robotics: Science and Systems XVI*. Robotics: Science and Systems Foundation, July 2020.
[17] M. Li, Dylan P. Losey, Jeannette Bohg, and Dorsa Sadigh. Learning user-preferred mappings for intuitive robot control. *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 10960–10967, 2020.
[18] Shaunak A. Mehta, Sagar Parekh, and Dylan P. Losey. Learning latent actions without human demonstrations. In *2022 International Conference on Robotics and Automation (ICRA)*, page 7437–7443. IEEE Press, 2022.
[19] Yash Chandak, Georgios Theocharous, James Kostas, Scott M. Jordan, and Philip S. Thomas. Learning action representations for reinforcement learning. *CoRR*, abs/1902.00183, 2019.
[20] Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. In *Conference on Robot Learning*, 2020.
[21] Zhanpeng He and Matei Ciocarlie. Discovering synergies for robot manipulation with multi-task reinforcement learning. In *2022 International Conference on Robotics and Automation (ICRA)*, page 2714–2721. IEEE Press, 2022.
[22] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.
[23] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J. Cree. Regularisation of neural networks by enforcing Lipschitz continuity. *Mach Learn*, 110(2):393–416, February 2021.
[24] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. volume abs/1609.09106, 2016.
[25] Sayantan Auddy, Jakob Hollenstein, Matteo Saveriano, Antonio Rodríguez-Sánchez, and Justus Piater. Continual learning from demonstration of robotics skills. *Robotics and Autonomous Systems*, 165:104427, 2023.
[26] Sami Haddadin, Sven Parusel, Lars Johannsmeier, Saskia Golz, Simon Gabl, Florian Walch, Mohamadreza Sabaghian, Christoph Jähne, Lukas Hausperger, and Simon Haddadin. The franka emika robot: A reference platform for robotics research and education. *IEEE Robotics & Automation Magazine*, 29(2):46–64, 2022.

VI. Appendix

We include the proofs for the theoretical results shown in Section III-C to explain our hypothesis on the user's expected experiences using nonliner action maps. We repeat the theorems here to make it more accessible to the reader.

Lemma 1. *For two states $x(t)$ and $\hat{x}(t)$, the action $a(t)$ under a Lipschitz continuous, odd, action map f and linearization $\hat{f} = \nabla_a f(\hat{x}(t), a)|_{a=0} a(t)$ at $a_0 = 0$ is $\|f(x(t), a(t)) - \hat{f}(\hat{x}(t), a(t)\| \leq M + LC$, where $M = \max_a \|f(x(t), a(t))\|$, L is the Lipschitz constant, and $C = \max\{\|a\| : a \in A\}$.*

Proof:

$$
\begin{aligned}
&\|f(x(t), a(t)) - \hat{f}(\hat{x}(t), a(t))\| \\
\leq &\|f(x(t), a(t)\| + \|\hat{f}(\hat{x}(t), a(t))\| \\
= &\|f(x(t), a(t)|| + \|\nabla_a f(\hat{x}(t), a)|_{a=0} a(t)\| \\
\leq &\|f(x(t), a(t)\| + \|\nabla_a f(\hat{x}(t), a)|_{a=0}\| \|a(t)\| \\
\leq &\|f(x(t), a(t)\| + L\|a(t)\| \leq M + L\|a(t)\| \\
\leq &M + LC. \ \square
\end{aligned}
$$

Theorem 1. *For an odd action map f that has Lipschitz constant L on a bounded action space, following Euler integration, the distance between trajectory $x(\tau)$ and $\hat{x}(\tau)$ where $\hat{X}$ follows linearization $\hat{f}$ of f, we have $\|x(\tau) - \hat{x}(\tau)\| \leq \nu(T-1)(M + LC)$, where ν is the discretization step and T is the number of discrete steps taken.*

Proof: Let $f_i = f(x_i, a_i)$, $\hat{f}_i = \hat{f}(\hat{x}_i, a_i)$, $E_i = \|x_i - \hat{x}_i\|$, and $\Delta f_i = \|f_i - \hat{f}_i\|$. It is possible to see that

$$
\begin{aligned}
&\|x_T - \hat{x}_T\| = \|x_{T-1} + \nu f_{T-1} - \hat{x}_{T-1} - \nu \hat{f}_{T-1}\| \\
\leq &\|x_{T-1} - \hat{x}_{T-1}\| + \nu \|f_{T-1} - \hat{f}_{T-1}\| \\
= &E_{T-1} + \nu \Delta f_{T-1} \\
\leq &E_{T-2} + \nu \Delta f_{T-2} + \nu \Delta f_{T-1} \\
\leq &\ ... \leq E_0 + \nu \sum_{i=1}^{T-1} \Delta f_i \quad \text{(Iterating the inequality above)} \\
\leq &\nu(T-1) \max_{i \in [1, T-1]} \Delta f_i \qquad (E_0 = 0 \text{ since } x(0) = \hat{x}(0)) \\
\leq &\nu(T-1)(M + LC). \ \square \qquad \text{(Lemma 1)}
\end{aligned}
$$