Curating User-Defined Interface Maps For Robot Teleoperation

Demiana R. Barsoum[1,2], Michelle H. Zhang[1], Larisa Y.C. Loke[1,2], and Brenna D. Argall[1,2]

***Abstract*— The interfaces used to operate assistive robots typically employ fixed, predefined maps to associate interface-level commands to robot control commands. User-defined control maps instead consider an individual's preferences and capabilities, moving away from a one-size-fits-all mapping paradigm. This work presents novel methods for (1) eliciting user-defined control maps, (2) identifying and addressing issues in control signal data that arise from such a user-centered design, and (3) methodologies to filter erroneous signals and construct synthetic data in an effort to address these issues. We experimentally evaluate our proposed methods by conducting a user study that elicits user-defined, interface-level commands for controlling a powered wheelchair and a robotic arm through four control interfaces. Our results highlight the differing suitability of user-defined, data-driven control maps for the various interface-platform pairings, and provide an analysis of the errors generated during dataset definition and the impact of our postprocessing methods.**

I. Introduction

Assistive robots—such as robotic arms and powered wheelchairs—play a crucial role in improving quality of life for many individuals living with motor impairments. By enhancing manipulation and mobility capabilities, these technologies advance user ability to complete daily functions with independence.

When an individual is fitted for an assistive device, an accompanying control interface—such as a joystick, head array, or sip/puff—is chosen and customized to best meet their needs and abilities. However, the limitations of such interfaces can render assistive devices difficult or unintuitive for users to control. A leading cause for the abandonment of assistive devices is lack of adaptability and configurability [1], which forces users to conform to interfaces and control schemes that do not adequately adapt to meet their individual needs. Research on device customization typically focuses on the interface hardware, including novel sensor setups and more diverse input options [2]. By contrast, less attention is given to the customization of control mappings used by interfaces that are the standards for assistive device control in clinical and real-world settings.

When a user actuates an interface, there is *noise* and *bias* associated with their issued control signals that is shaped by their physiology, psychology, and preferences. For example, handedness may result in biases in joystick actuation. Due to these noise and bias patterns, the ability of an individual to use an interface depends heavily on the alignment between their behavioral patterns and those expected by the control mapping. This dependence can restrict the accessibility of an interface for users whose behaviors do not align well with predefined expectations.

[1]Northwestern University, Evanston, IL., USA
[2]Shirley Ryan AbilityLab, Chicago, IL., USA

Interface control maps moreover can be unintuitive to operate, and require extensive practice before users are able to control devices effectively [3]. We posit that to investigate the noise and preference patterns in user control signal data is useful not only for identifying these patterns, but also for uncovering improved methods able to accommodate individual differences within the control interface design. Such an analysis ultimately might enable the creation of interfaces that are more accessible to a wider range of users.

In this paper, we propose and evaluate methods for constructing task-agnostic, user-defined control maps for robot teleoperation that adapt interface operation to the unique capabilities and preferences of a given user. This work makes the following contributions:

(1) We present a protocol for constructing datasets of task-agnostic interface use, for the purpose of generating customized interface maps for teleoperation.

(2) We design methods to identify, quantify, and filter artifacts that arise in datasets generated from user-defined, data-driven control interface design.

(3) We introduce an approach for generating synthetic data from users' verbal statements to capture their intended control mappings.

(4) We evaluate our proposed methods by conducting a participant study that elicits user-defined, interface-level commands for controlling a powered wheelchair and a robotic arm through four control interfaces (Fig. 1).

The rest of the paper is organized as follows. Section II outlines related literature. Section III details the proposed data processing methods. Section IV presents the user data collection pipeline, including experimental setup and protocol. Results and discussion are presented in Section V, including limitations and avenues for future work in Section VI, with conclusions presented in Section VII.

II. Background

In this section, we present a summary of related literature on teleoperation control and customization.

A. Control Interfaces and User Customization

Assistive devices and robots can be teleoperated using a variety of accessible interfaces such as joysticks, sip/puffs, eye gaze trackers, head arrays, brain-computer interfaces, body-movement interfaces, tongue-controlled interfaces, and lip-controlled interfaces, as well as customized setups and multi-interface systems.

Customization of these interfaces is a widely researched topic [4]–[6]. Joystick handles, for example, can be re-

designed to allow control using wrist movements [7], accommodating users with limited or no hand motion. Interfaces such as the sip/puff and MouthPadˆ (Augmentalˆ, California, USA) are designed for individuals with upper body paralysis. While their actuation mechanisms are accessible, the learning curve for using accessible interfaces to operate assistive devices and robots can be high [5], particularly as many issue lower-dimensional signals than the device being controlled and, as a result, use modal control paradigms (mode switching) [8].

B. Control Mappings for Teleoperation

Teleoperation is an attractive means of controlling assistive devices because it allows users to remain maximally in control, preserving user agency. However, the mismatch between a user's physical capabilities and the input requirements of standardized control mechanisms can hinder effective device operation, leading to frustration, fatigue [4], and device abandonment [1]. Work in the design of control maps that explicitly aim to address such a mismatch is overviewed here.

In detail, the mapping from interface actuation signal ϕ (e.g., joystick deflection) to machine control command u (e.g., joint or end-effector velocity) is typically static and defined *a priori* by experts (system engineers or assistive technology technicians). There is work that investigates instead learning user-preferred mappings for intuitive robot control [9], where the learned mapping is task-dependent. Customizing standard control maps to accommodate users' characterized physiological variability is also explored [10], using an approach that relies on a predefined baseline map. Other work aims to explicitly model discrepancies between intended ϕ_i and measured ϕ_m interface commands [11], where discrepancies might arise due to physiological factors such as fatigue or motor impairment, or an incomplete understanding of the robot system. All of these works evaluate a single combination of interface and platform.

In this paper, we propose a methodology for the design of *user-defined* control maps that are *task-agnostic* and leverage knowledge of intended interface commands—by capturing user biases and preferences— to postprocess noisy and scarce data. We also evaluate this methodology on multiple interfaces, operating two robot platforms.

III. Technical Methods

In this section, we provide an overview of our proposed methods for constructing datasets for the purpose of learning user-defined interface maps.

We first present a methodology to elicit users' preferred interface commands for task-agnostic robot actions (Sec. III-A). However, a dataset gathered from examples of human-issued interface commands and associated robot motion labels can be prone to data sparsity and inconsistencies between the human's intended versus issued interface commands (Sec. III-B). We therefore propose a suite of data processing tools that aim to address these potential pitfalls (Sec. III-C and Sec. III-D). In particular, our methods leverage verbal descriptions from the user of their intended interface commands, which we convert to automated scripts

Fig. 1: Experimental Hardware. Robot platforms (*top, left to right*): Sunrise Quickie powered wheelchair and Kinova JACO robotic arm. Control Interfaces *(bottom, left to right)*: 3-axis Joystick, Sip/Puff, 3-switch Head Array, and x-IMU3 sensors.

that generate both *filtered data* that refines the raw signals and *synthetic data* based on the user's vocalized intent.

A. User-Defined Control Interface Maps

We consider a user-defined interface-level command to represent the user's intended actuation of an interface to produce an observed robot motion. These commands can be continuous, discrete, or both, depending on the control interface. Continuous signals can be interpreted proportionally, while discrete signals, such as button presses, only provide an indicator of activation without magnitude. An "intuitive" control map has previously been defined [9] to have the following properties:

- Proportional: The system's response scales with the command's input intensity.
- Reversible: Opposing interface actuations (e.g., left-right deflection) correspond to inverse commands.
- Consistent: Users can remember the correspondence between interface actuations and system controls, and reliably issue them.

In this work, we prompt users to issue their preferred interface-level command ϕ in response to observed robot motion a. We furthermore propose a data collection protocol that *annotates* this data with vocalizations of their intended interface-level commands—capturing complete, unfiltered interface interactions, including noise and incorrectly issued (unintended) commands that may diminish the quality of the data. The purpose of these intent annotations is to provide a means for repairing such artifacts.

While such an approach leaves the control design in the hands of the user, it can also result in violations of the aforementioned properties of an intuitive control map, as well as artifacts in the data that pose considerable challenges for model training. These challenges ultimately inform our data postprocessing and synthetic data generation methods (Secs. III-C, III-D).

B. Implications of User Design

The usage characteristics of a given interface, combined with the user's understanding of its control method, leave

TABLE I: Discrepancies observed in user-defined datasets and proposed methods to handle these issues.

Artifact	Sources	Data Processing Techniques
Incorrect Command	(1) Unfamiliar with interface (2) Struggle to actuate interface (3) Forget how to issue a command or how to actuate the interface	(a) Replace with synthetic data (b) Replace with correct data from user (c) Remove from dataset
One-to-Many Command	(1) Issue similar (or identical) signals for different robot actions	(a) Replace with synthetic data
Data Sparsity	(1) Insufficient training data for command (2) Similar signals for different robot actions	(a) Add synthetic data (b) Add copies of raw user data
Noisy Signal	(1) User actuation behavior (2) Environmental interference	(a) Replace with synthetic data (b) Replace with correct data from user

a distinct imprint on the signal intended for robot control. This imprint can be beneficial if it helps produce a control mapping tailored to the user's unique behavior, or it can be problematic if the user develops inefficient or inconsistent control patterns. Additionally, the inherent variability in allowing users to design their own control maps may result in unusual or unpredictable control signals.

In this work, we identify four *categories of user-induced dataset artifacts*, along with their potential sources, summarized in Table I. We highlight potential strategies for mitigating issues that arise from these artifacts in the *Data Processing Techniques* column.

Furthermore, we identify and define five notable *types of actuation error* that can result from these artifacts and contribute to the aforementioned dataset challenges. To do so, we first define an interface channel to be an independent input signal of a control interface. Each channel is activated through an actuation mechanism: for example, joystick deflection, head array button press, or sip/puff respiration. Channels can be discrete or continuous in value. The types of actuation error then are defined as follows:

(1) Incorrect Activation: At least one intended channel is activated but in the wrong direction, or at least one unintended channel is activated with a magnitude greater than or equal to that of the intended channel(s).

(2) Additional Channel Activation: The intended channel(s) is activated, but there are also nonzero signals in one or more extraneous channels with magnitudes lower than those of the intended activations.

(3) Missing Channel Activation: The intended channel is not activated for a substantial portion of the command duration.

(4) Transient Zero: At least one intended channel momentarily reports zero activation when it is expected to have a consistent nonzero value.

(5) Leading/Trailing Zero: An intended channel exhibits periods of zero activation, as a result of the user pausing before and/or after issuing a command.

In our implementation, the groundtruth intended channels are derived (Sec. III-C) from a user's vocalized annotations.

C. Methodology for Filtering

Here, we present our methodology for identifying and filtering out the aforementioned dataset artifacts based on the error type. We propose a ***filtering methodology*** that *removes* instances of:

(1) Incorrect activation.

(2) No channel activation, which includes missing channel activations, transient zeros, and leading/trailing zeros.

In order to determine signal correctness, we leverage user annotations of intended interface signals. Specifically, our data collection protocol (Sec. III-A) annotates examples of interface→robot control mappings with vocalizations of the intended interface command. Our application of this filtering methodology is coarse and applies only to:

(1) Signal *absence* or *incorrect sign* on intended channels.

(2) Signal *presence* on unintended channels.

Notably, such an application may not catch all errors—incorrect patterns of interface activations, for example, will be missed. However, the likelihood that it filters correct signals will also be minimized.

D. Methodology for Synthetic Data Generation

Next, we present our methodology for generating synthetic data to rectify dataset artifacts. We propose a ***synthetic data generation methodology*** that *replaces* instances of:

(1) Signal *absence* or *incorrect sign* on intended channels.

Notably, this replacement approach retains unintended channel activations, as well as commands whose magnitudes differ from those of the intended signals, in order to preserve real-world variability and noise to enrich the dataset.

For a user annotation to be eligible for synthetic data generation, it must meet the following criteria:

(1) Non-varying in magnitude.

(2) No intended transient zeros (patterns).

As with our filtering methodology, such an application will not generate data for all annotations: only straightforward and unambiguous intended inputs will qualify. We leave the interpretation of more complex annotations and input patterns to future work.

To generate synthetic data, we define replacement rules for the three most common interfaces used to operate powered wheelchairs: joystick, head array, and sip/puff. We build these synthetic representations by hand, according to the following prescription:

(1) **Joystick Mapping**: The joystick interface utilizes a three-axis control scheme, denoted as $[x, y, z]$. Based on the deflection direction, the joystick mapping translates to the corresponding channel activations as follows:

- **Forward**: [0.0, 1.0, 0.0]
- **Backward**: [0.0, -1.0, 0.0]
- **Left**: [1.0, 0.0, 0.0]

- **Right**: [-1.0, 0.0, 0.0]
- **Twist Left (Counterclockwise)**: [0.0, 0.0, 1.0]
- **Twist Right (Clockwise)**: [0.0, 0.0, -1.0]

(2) **Head Array Mapping**: The head array interface consists of multiple pads that once pressed map to specific channel activations:

- **Back Pad**: [1, 0, 0, 0]
- **Right Pad**: [0, 1, 0, 0]
- **Left Pad**: [0, 0, 1, 0]
- **Chin Switch**: [0, 0, 0, 1]

(3) **Sip/Puff Mapping**: The sip/puff interface maps to continuous values based on respiration pressure. The range for a user is based on their intent vocalizations:

- **Sipping**: Ranges from 0.0 to 1.0
- **Puffing**: Ranges from -1.0 to 0.0

These rules are applied to data that meet the replacement criteria defined above.

IV. Experimental Procedure

In this section, we overview the details of our evaluation study in which participants observe the movements of a robotic wheelchair and a robotic arm, and are asked to vocalize, practice, and issue their intended interface-level commands corresponding to the movements.

A. Participants

A total of ten participants without motor impairment (6 female, 4 male) and with ages ranging from 23 to 52 years (mean 30.1$\pm$8.18) were recruited for this study. Participants gave their informed, signed consent to participate in the experiment, which was approved by Northwestern University's Institutional Review Board (STU00217536). None of the participants had prior experience with powered wheelchair driving. One participant (F, 25) had experience operating robotic arms, but not the robotic arm used in this study.

Each volunteer participated in two study sessions, during which a single robot platform was operated. The order of platform presentations was random and approximately balanced across participants (6 of 10 interacted with the robotic wheelchair in the first session).

B. Hardware

Two robotic platforms and a total of four control interfaces are utilized in the study (Fig. 1).

Robot Platforms. The first robot platform is a robotic wheelchair, built atop a Sunrise (Arizona, USA) Quickie Q500M base. Platform control is achieved through the LUCI ROS 2 SDK [12]. The wheelchair is a differential drive robot, which operates in 2-D, allowing for translational (forward/backward) and rotational (spinning) movement.

The second robot platform is a Jaco Gen2 robotic arm from Kinova Robotics (Quebec, Canada). Platform control is achieved through the Kinova SDK [13]. The Jaco is a 7-DoF non-anthropomorphic robotic arm with a 3-finger gripper—offering 6-D control (3-D translation and 3-D rotation) in task space.

Participants have no control over the motion of the robots during the study. All movements (Sec. IV-C) are prerecorded and controlled by the experimenters.

Control Interfaces. Three of the four interfaces—joystick, sip/puff, and switch-based head array—are commercial standards for powered wheelchair operation, selected to encompass a broad range of dimensionality, continuity, and actuation mechanisms. The fourth interface is not a commercial standard, but offers a richer input space to better match the higher-dimensional output space of a robotic arm. It uses a set of n inertial measurement unit sensors (IMUs) to track the upper body motions of participants, where n is determined based on the movements chosen by the participant (e.g., two-arm versus one-arm motions).

- Joystick. A 3-axis joystick (CH Products, California, USA) with 3 axes of deflection and 12 buttons.
- Sip/Puff. A 1-D USB Sip/Puff Breeze Switch (Origin Instruments, Texas, USA) that is operated via respiration and outputs a continuous-valued pressure signal.
- Head Array. A 3-switch FUSION Proportional and Digital head array with a chin switch (Adaptive Switch Laboratories, Inc., Texas, USA) that detects head taps to the back, left, and right pads and chin presses to the chin switch. This interface is used with the ASL ATOM Wireless Mouse Emulator which converts pad and button activations into discrete mouse inputs for data recording on a computer.
- IMU Sensors. The IMU-based motion tracking interface consists of x-IMU3 sensors (x-io Technologies Limited, Bristol, UK) attached to the participant via x-IMU3 elasticated straps. Each IMU outputs a 4-D (quaternion) velocity signal, corresponding to the motion of the participant body part to which it is affixed.

Participants are free to issue commands over any number and combination of channels on the given interface in use.

C. Study Protocol

Setup for Wheelchair. During the powered wheelchair session, each participant is seated in the wheelchair and asked to physically actuate the joystick, head array, and sip/puff interfaces. The head array is positioned for optimal activation. Participants observe 8 prerecorded, task-agnostic robot motions: *forward, backward, clockwise-spot-turn, counterclockwise-spot-turn, forward-clockwise, forward-counterclockwise, backward-clockwise,* and *backward-counterclockwise.*

Setup for Robotic Arm. During the robotic arm session, each participant is seated a safe distance from the arm and asked to physically actuate the joystick, sip/puff, and IMU interfaces. Participants are presented with 12 prerecorded, task-agnostic robot motions: *forward, backward, left, right, up, down, roll-clockwise, roll-counterclockwise, pitch-up, pitch-down, yaw-left*, and *yaw-right*.

Study Procedure. The study consists of two phases that are repeated for each interface:

(1) *Familiarization*. Participants are instructed on all avail-

able channel activations of the given interface.[1]

(2) *Map Data Collection*. Participants observe the prerecorded robot motions, presented as reversible pairs (e.g., forward/backward) in a randomized order that is balanced across participants. (The presentation order within the pairs themselves is also randomized and balanced.) For each observed motion:

- Participants first verbalize the interface-level command they would like to associate with the observed motion and are encouraged to practice issuing this command through the interface.
- Participants issue the interface-level command they have verbalized. To help anchor their command, the prerecorded robot motion is repeated as they issue their intended command through the interface.

Control signal data from the interface is recorded during both steps, including practice. For the sip/puff interface, visual feedback in the form of a pressure signal plot is provided.

D. Datasets

We generate *six* datasets based on our data processing methodologies presented in Section III:

- *Raw*: User's issued interface actuation signals.
- *Filtered*: Refines the raw signals according to the methodology described in Section III-C.
- *Synthetic*: Contains data generated based on the user's intended interface commands according to the methodology described in Section III-D.
- *Synth-raw*: Union of raw and synthetic datasets.
- *Synth-filtered*: Union of filtered and synthetic datasets.
- *Balanced*: Upsamples the raw signals to ensure all classes have the same number of samples as the most populated class in that raw dataset.

The *Synth-raw* formulation expands the size of the dataset while preserving the richness of the raw user interface-level commands. By retaining the inherent noise of the raw signals, the machine learning model is encouraged to tolerate variability and identify underlying patterns within the data. By contrast, the *Synth-filtered* formulation removes this noise.

We also compare the performance of our synthetic data generation method against state-of-the-art techniques:

- *SoTA-cont:* Random noise augmentation [14] by adding Gaussian noise to the raw signals ($\mu = 0, \sigma^2 = 0.05$).
- *SoTA-disc:* A Conditional Generative Adversarial Network (CGAN) [15].

While random noise augmentation is broadly deployed throughout the field, it is not suitable for discrete inputs. We therefore train a CGAN on the set of $\{(\phi_m^t,\ a^t)\}$ pairs for the head array interface. By conditioning both the generator and discriminator on a^t, the CGAN is able to produce meaningful synthetic data that mirrors the original dataset and is contextually relevant to the robot's actions.

We thus generate seven datasets (six based on our methodologies and one SoTA) for each interface-robot combination, with one notable exception: the IMU Sensors. None of the data generated using the IMU Sensors interface met the eligibility criteria for our Filtering or Synthetic Data Generation methodologies (Sec. III-C and III-D). Thus, only *raw* datasets are generated for this interface.

E. Machine Learning Algorithms

We select a subset of open-source machine learning algorithms to assess the viability of our proposed dataset generation methods using the open-source package PyCaret [16], which uses Scikit-learn [17] for the implementation and automated hyperparameter tuning of multiple algorithms.

Given the structure of our data, $(\phi_m^t,\ a^t)$, which includes robot motion labels, we focus on a supervised learning approach. We frame our problem as a classification task, requiring each model to predict a probability distribution over robot action labels given an interface signal input, and outputting the highest probability action. We explore a total of 16 machine learning algorithms, ranging from simple and lazy such as K-Nearest Neighbors (KNN) to more complex architectures such as Neural Networks (NN)—plus a chance classifier baseline that predicts a random label. Our selection spans a diverse range of capabilities, including models suited for high-dimensional data (e.g., Gaussian Naive Bayes [18]) and those effective for handling complex, high-variance data (e.g., ensemble methods [19]).

We implement a Feed-Forward Neural Network using TensorFlow and Keras [20], while our CGAN is implemented in PyTorch [21]. The hyperparameters for these two models, which are not included in PyCaret, are detailed in Table II.

To build and validate our models, we perform an 80/20 split into training and testing sets for each dataset. We conduct 10-fold cross-validation on the training set and take the best-performing of these to be our model, which is then evaluated on the holdout test set data. The results in Section V report on this test set performance.

TABLE II: Neural Network and CGAN model parameters and training hyperparameters.

Machine Learning Model	**Hyperparameters**	
Neural Network (NN)	activation function	relu & softmax
	loss function	CCE
	learning rate	0.001
	epoch	200
	optimizer	'Adam'
Conditional Generative Adversarial Network (CGAN) (Generator only)	activation function	relu & softmax
	loss function	BCE
	learning rate	0.001
	epoch	5000
	optimizer	'Adam'

F. Analysis Methods

To evaluate the performance of our models across the different datasets, we use both entropy and accuracy. *Entropy*, H, quantifies the level of uncertainty in a prediction. Lower entropy values indicate reduced uncertainty in the model. The

[1]For robotic arm sessions, prior to interface familiarization, participants additionally are presented with the prerecorded robot arm motions (which are more complex and unfamiliar).

following equation [22], modified to include normalization, is used to calculate H:

$$H_{\text{norm}}(X) = -\frac{1}{\log_2(n)} \sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

where n is the number of class labels for each robot platform.

For each prediction, we compute H on the probability estimates for the most likely class label, classifying the prediction into one of four categories: (a) correct and certain, (b) wrong and certain, (c) correct and uncertain, and (d) wrong and uncertain. We define a second performance metric, *certain-accuracy*, which accounts for both model certainty and accuracy, using the formula:

$$\textit{Certain-Accuracy} = \frac{|a|}{|a| + |b|}$$

We define a baseline threshold for whether a model is predicting with certainty to be the entropy of a random classifier, which varies by participant-platform-interface-dataset type combination. All models' entropy values are compared to this threshold.

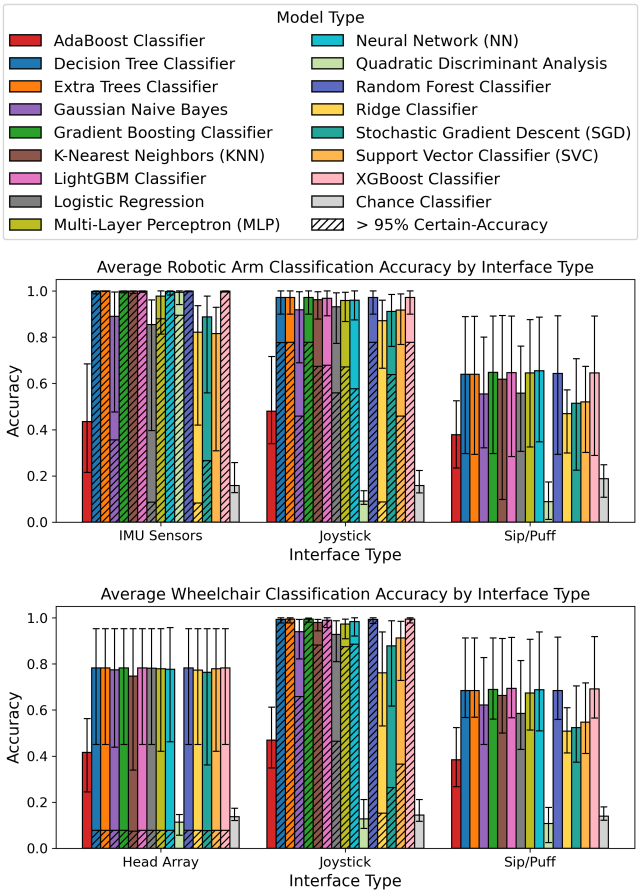

Fig. 2: Average model accuracy across all participants (S01-S10) for the robotic arm (top) and robotic wheelchair (bottom) platforms, clustered by interface type. For each bar (interface-platform-model combination), the proportion of participants whose model demonstrated a certain-accuracy greater than 95% is indicated with hatch lines.

V. Results and Discussion

We first evaluate the performance of a suite of machine learning models trained and tested on the raw datasets. For models that fall below a performance threshold of 60% prediction accuracy, we analyze the impact of our postprocessing methodologies. We also quantify the types of actuation errors (Sec. III-C) observed in the raw datasets.

A. Performance of Models

For each classification algorithm, Figure 2 shows the average accuracy aggregated across all 10 participants grouped by interface type for each robotic platform. These results indicate that multiple algorithms are able to successfully learn control map models for the joystick and IMU sensors interfaces, while all algorithms struggle to learn models for the sip/puff and head array interfaces. This struggle is evident in both the overall accuracy and the certain-accuracy demonstrated by the resulting models.

Notably, the sip/puff and head array interfaces offer less complex—lower-dimensional and/or discrete—input signals, which introduces challenges such as data sparsity and one-to-many command mappings (if participants develop difficult-to-disambiguate patterns to compensate for limited input options). By contrast, the IMU sensors interface offers the richest input signals and achieves the highest performance of all interface-platform combinations, with multiple algorithms demonstrating perfect accuracy.

Table III summarizes the algorithms that are high-performing on the raw datasets.[2] Here, the determination of *high performance* is a prediction accuracy that exceeds the 95% threshold, when averaged across all participants.

Of these 10 algorithms, 9 are high-performing on half of the interface-platform combinations. Among these, 6 are ensemble methods—Decision Tree, Extra Trees, Gradient Boosting, LightGBM, Random Forest, and XGBoost—which combine predictions from multiple weak models with high variance to produce a single prediction with low variance. Such an approach appears to be well-suited to the high variability, low sample volume, and sparsity present in our datasets. The remaining algorithms that are high-performing but are *not* ensemble methods are the Neural Network, Multi-Layer Preceptron (MLP), and K-Nearest Neighbor algorithms, all of which are able to model highly nonlinear decision boundaries.

B. Analysis of Methodologies

We analyze the impact of our postprocessing methodologies on *prediction accuracy* and *prediction certainty* for participants whose (*raw*) models are low-performing, meaning they achieve less than 60% prediction accuracy. For raw datasets that are already high-performing, postprocessing methodologies are of limited utility. We assess prediction certainty through a measure of entropy, where lower entropy correlates with lower uncertainty. Two platform-interface combinations meet this low performance criteria: **robotic arm & sip/puff** (participants S04, S08, and S09) and **wheelchair & head array** (participants S05 and S10).

[2]Four algorithms are never high-performing on any dataset, so they are omitted from this table: AdaBoost, Gaussian Naive Bayes, Ridge Classification, and Stochastic Gradient Descent (SGD).

TABLE III: High Performance Algorithms on the Raw Datasets.

Platform	Interface	Decision Tree	Extra Trees	Gradient Boosting	KNN	LightGBM	MLP	NN	Quadratic Discriminant Analysis	Random Forest	XGBoost
Robotic Arm	IMU Sensors	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Robotic Arm	Joystick	✓	✓	✓	✓	✓	✓	✓		✓	✓
Robotic Arm	Sip/puff										
Wheelchair	Head Array										
Wheelchair	Joystick	✓	✓	✓	✓	✓	✓	✓		✓	✓
Wheelchair	Sip/puff										

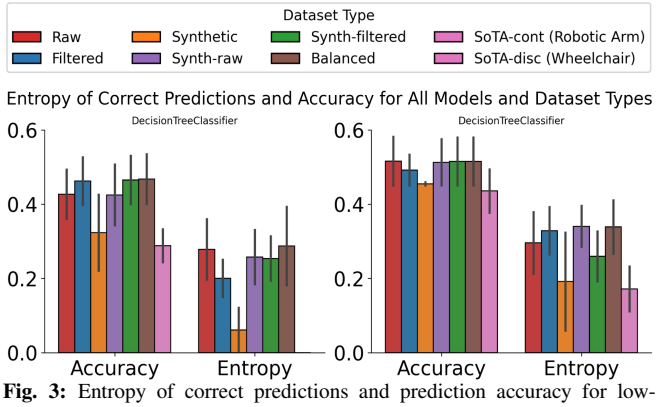

Fig. 3: Entropy of correct predictions and prediction accuracy for low-performance (lower than 60% accuracy) models trained and tested on the raw datasets: robotic arm & sip/puff (*left*, participants S04, S08, and S09) and wheelchair & head array (*right*, participants S05 and S10).

Only these five participant-platform-interface combinations are evaluated in the following postprocessing methodology analysis.

Figure 3 presents the entropy of correct predictions and prediction accuracy for each low-performance platform-interface combination, averaged over the five participants. For presentation clarity, only one representative algorithm is shown (Decision Tree Classifier); results for all algorithms are summarized next.

For the robotic arm & sip/puff combination shown in Figure 3, only one methodology noticeably reduces entropy without a sacrifice in prediction accuracy (in fact, accuracy slightly improves): our *Filtered* methodology. Across all tested machine learning algorithms, 62.5% (10 of 16) exhibit this trend of decreasing entropy while preserving or slightly increasing accuracy with the *Filtered* methodology, in comparison to the raw dataset. Additionally, 10 (of 16) exhibit this trend with the *Synth-filtered* methodology, and 1 (of 16) with *Synth-raw*.

Any other methodologies that reduce entropy do so at the cost of prediction accuracy, while other methodologies that preserve prediction accuracy do so without reducing entropy. Notably, the SoTA (*SoTA-cont*) and the standard data postprocessing technique of upsampling (*Balanced*) exhibit this trend for only *one* algorithm; in all other cases, our postprocessing methodologies outperform both the SoTA and standard postprocessing technique.

For the wheelchair & head array combination shown in Figure 3, it is also the case that only one methodology noticeably reduces entropy without a sacrifice in prediction accuracy: our *Synth-filtered* methodology. This trend again holds across 62.5% (10 of 16) of the tested machine learning algorithms. Additionally, 2 (of 16) exhibit this trend with the *Synthetic* methodology, and 1 (of 16) with each of *Filtered* and *Synth-raw*.

Notably, for this platform-interface combination, *neither* the SoTA (*SoTA-disc*) nor the standard data postprocessing technique of upsampling (*Balanced*) ever exhibit this trend for *any* algorithm. Our collective postprocessing methodologies thus outperform both the SoTA and standard postprocessing technique.

C. Quantifying Errors

For interfaces that meet our criteria for filtering (joystick, sip/puff, head array), we quantify actuation errors as defined in Section III-B. Errors are tallied for each participant, and the aggregated counts are reported in Figure 4. An illustration of selected individual participant error counts and types is also provided for the robotic arm & joystick combination—showcasing the variability in errors across participants. Note that a single actuation can contribute to multiple error categories (e.g., both "wrong command" and "missing activation").

The most frequent errors across all interfaces are **incorrect activation** and **leading/trailing zeros**. As expected, given the challenges of using the sip/puff device, **incorrect activation** errors are more prevalent for this interface. For head array, a notable number of **missing channel** errors are observed, likely due to suboptimal device positioning that can hinder a user's ability to issue simultaneous commands.

VI. Limitations and Future Work

A major limitation of this work is that our method was evaluated solely using data from people without motor impairment. In practice, motor impaired users may experience interface actuation difficulties unique to their conditions that were unobserved in the data we collected, and could affect the performance of our methodologies.

Future work will extend this work to end-users, evaluating how user-defined interface maps can be customized to their specific interfaces, needs, and preferences. We will also evaluate the utility of these user-defined control maps when used for robotic teleoperation. For the sip/puff interface, where a clear method for filtering data is currently lacking due to user variations in definitions for Φ, future efforts will explore better methods to filter noise both online (real-time) and offline (post-hoc), as well as investigate practical strategies for synthesizing data. While user-driven design is important, due to artifacts that can arise, a more collaborative design process will also be explored where researchers provide feedback on the feasibility of preferred commands.

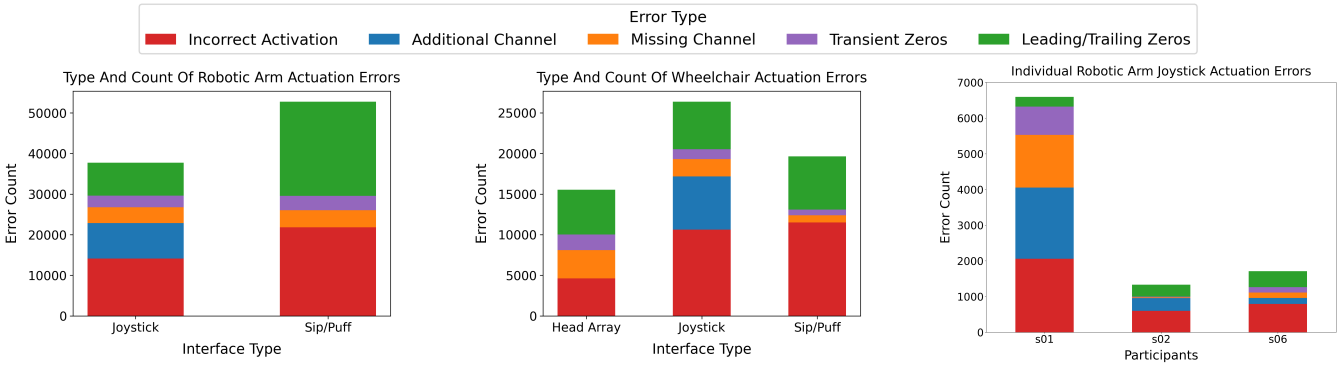

Fig. 4: Aggregated counts and types of actuation errors in the raw dataset for each platform-interface combination meeting the filtering criteria (Sec. III-C), tallied across all participants (left and center) and showcasing the variability between individual participants (right).

VII. Conclusions

In this work, we presented methods for curating user-defined interface control maps, addressing challenges in user-centered design, and filtering actuation errors to enhance dataset quality. Our experimental work evaluated, in a participant study, user-defined interface maps for two robotic platforms and four control interfaces. The performance of a suite of machine learning models on this data showed that multiple algorithms were able to successfully learn maps derived from raw joystick and IMU sensor signals. For platform-interface combinations that struggled to learn mappings between interface-level commands and robot actions, training on datasets curated via our postprocessing methodologies was demonstrated to decrease uncertainty. These results underscore the differing suitability of user-defined, data-driven control maps for various interface-platform combinations. By enhancing adaptability and personalization in assistive device control, this work contributes to more user-customized robotic teleoperation systems.

Acknowledgment

The authors would like to thank Corinne Fischer, OTR/L, Occupational Therapist at the Shirley Ryan AbilityLab, for assisting with sessions. Funding for this work was provided by the National Science Foundation under Grant CMMI-2208011. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] A. T. Sugawara, V. D. Ramos, F. M. Alfieri, and L. R. Battistella, "Abandonment of assistive products: assessing abandonment levels and factors that impact on it," *Journal of Disability and Rehabilitation: Assistive Technology*, vol. 13, no. 7, p. 716–723, 2018.

[2] C. L. Fall *et al.*, "Wireless semg-based body–machine interface for assistive technology devices," *Journal of Biomedical and Health Informatics*, vol. 21, no. 4, p. 967–977, 2016.

[3] Y. M. L. R. Silva, W. C. S. S. Simões, E. L. M. Naves, T. F. Bastos Filho, and V. F. De Lucena, "Teleoperation training environment for new users of electric powered wheelchairs based on multiple driving methods," *IEEE Access*, vol. 6, 2018.

[4] I. Mougharbel, R. El-Hajj, H. Ghamlouch, and E. Monacelli, "Comparative study on different adaptation approaches concerning a sip and puff controller for a powered wheelchair," in *Proceedings of the IEEE Science and Information Conference*, 2013.

[5] H. S. Grewal, A. Matthews, R. Tea, V. Contractor, and K. George, "Sip-and-puff autonomous wheelchair for individuals with severe disabilities," in *Proceedings of the IEEE Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2018.

[6] S. Pozzi and S. Bagnara, "Individuation and diversity: the need for idiographic hci," *Journal of Theoretical Issues in Ergonomics Science*, vol. 14, no. 1, p. 1–21, 2013.

[7] H. D. Shin *et al.*, "Customized power wheelchair joysticks made by three-dimensional printing technology: A pilot study on the environmental adaptation effects for severe quadriplegia," *International Journal of Environmental Research and Public Health*, vol. 18, p. 7464, 2021.

[8] L. V. Herlant, R. M. Holladay, and S. S. Srinivasa, "Assistive teleoperation of robot arms via automatic time-optimal mode switching," in *Proceedings of the International Conference on Human-Robot Interaction (HRI)*, 2016.

[9] M. Li, D. P. Losey, J. Bohg, and D. Sadigh, "Learning user-preferred mappings for intuitive robot control," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[10] A. Thompson, L. Y. Loke, and B. Argall, "Control interface remapping for bias-aware assistive teleoperation," in *Proceedings of the IEEE International Conference on Rehabilitation Robotics (ICORR)*, 2022.

[11] M. N. Javaremi, L. Y. Loke, and B. Argall, "Experimental validation of interface-aware assistance for 7-dof robot arm teleoperation," in *International Symposium on Experimental Robotics*, 2023.

[12] C. Prather, A. Ernst, and R. Dasgupta, "luci-ros2-sdk," https://github.com/lucimobility/luci-ros2-sdk.git, 2022.

[13] Kinova, "kinova-ros," https://github.com/Kinovarobotics/kinova-ros.git, 2022.

[14] M. Momeny *et al.*, "Learning-to-augment strategy using noisy and denoised data: Improving generalizability of deep cnn for the detection of covid-19 in x-ray images," *Journal of Computers in Biology and Medicine*, vol. 136, p. 104704, 2021.

[15] M. Mirza, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[16] M. Ali *et al.*, *PyCaret: An open source, low-code machine learning library in Python*, 2020, pyCaret version 1.0.0.

[17] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] L. Chen and S. Wang, "Automated feature weighting in naive bayes for high-dimensional data classification," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 1243–1252.

[19] D. Pedro, "A unified bias-variance decomposition and its applications," in *Proceedings of the 17th International Conference on Machine Learning*, 2000, pp. 231–238.

[20] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[21] J. Ansel *et al.*, "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2024.

[22] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, 1948.