# PECAN: Personalizing Robot Behaviors through a Learned Canonical Space

HERAMB NEMLEKAR, ROBERT RAMIREZ SANCHEZ, and DYLAN P. LOSEY, Virginia Tech, USA

Robots should personalize how they perform tasks to match the needs of individual human users. Today's robot achieve this personalization by asking for the human's feedback in the task space. For example, an autonomous car might show the human two different ways to decelerate at stoplights, and ask the human which of these motions they prefer. This current approach to personalization is *indirect*: based on the behaviors the human selects (e.g., decelerating slowly), the robot tries to infer their underlying preference (e.g., defensive driving). By contrast, our paper develops a learning and interface-based approach that enables humans to *directly* indicate their desired style. We do this by learning an abstract, low-dimensional, and continuous canonical space from human demonstration data. Each point in the canonical space corresponds to a different style (e.g., defensive or aggressive driving), and users can directly personalize the robot's behavior by simply clicking on a point. Given the human's selection, the robot then decodes this canonical style across each task in the dataset — e.g., if the human selects a defensive style, the autonomous car personalizes its behavior to drive defensively when decelerating, passing other cars, or merging onto highways. We refer to our resulting approach as PECAN: **Pe**rsonalizing Robot Behaviors through a Learned **Can**onical Space. Our simulations and user studies suggest that humans prefer using PECAN to directly personalize robot behavior (particularly when those users become familiar with PECAN), and that users find the learned canonical space to be intuitive and consistent. See videos here: https://youtu.be/wRJpyr23PKI

## 1 INTRODUCTION

Over its lifetime, a robot will likely interact with multiple different humans. Each of these humans has their own personal preferences for how the robot should behave. For example, consider an autonomous car that drives a human passenger. One passenger might prefer for the autonomous car to be especially defensive, while another passenger may want the car to drive more aggressively. In order to account for this personalization, we recognize that it is not sufficient for a robot to just learn the desired tasks it should perform. We also need robots that adapt the *way they perform those tasks* (i.e., adapt their *style*) to match the current user's preferences.

Existing research often tries to address this problem by collecting human feedback in the task space. Here the human can demonstrate their desired trajectory, correct the robot's motion, or rank the robot's behavior [3, 12, 17, 20–22, 25, 27, 30, 36]. The robot then uses this feedback to update its estimate of what style the human truly wants: for example, if a human passenger shows the autonomous car that it should gradually decelerate at red lights, the autonomous car might infer that the human prefers defensive driving. Unfortunately, this task-space approach is fundamentally limited because it results in *indirect personalization.* The human user is not able to directly convey their desired style (e.g., defensive driving). Instead, the human must show behaviors that exhibit their desired style in the task space (e.g., gradually decelerating), and hope that the robot infers the correct style from this data.

By contrast, in this paper we enable humans to *directly personalize* the robot's behavior across multiple tasks with a single click. We achieve this shift by lifting human-robot interaction from the task space into the style space. More specifically, we hypothesize that:

*Styles are often shared across tasks, and these*

This work is supported in part by NSF Grant #2246446.
Authors' address: Heramb Nemlekar, hnemlekar@vt.edu; Robert Ramirez Sanchez, robetjrs03@vt.edu; Dylan P. Losey, losey@vt.edu, Virginia Tech, Department of Mechanical Engineering, 635 Prices Fork Rd, Blacksburg, VA, 24060, USA.

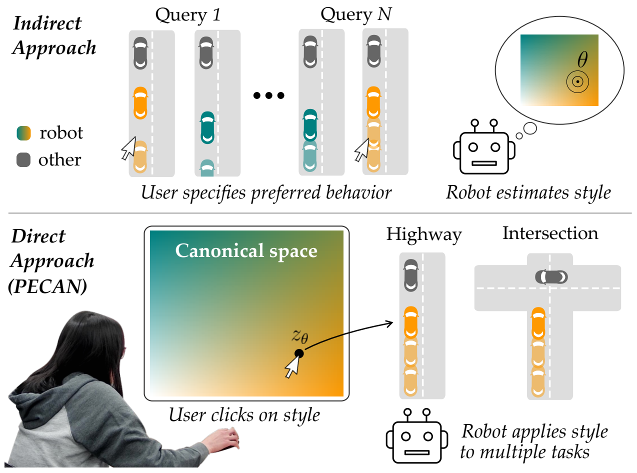

Fig. 1. User personalizing the driving style of an autonomous car. With existing methods, the user provides feedback about their preferred behavior, and the robot *indirectly estimates* their style based on this feedback. We propose a *direct* approach where users select their style in a canonical space, and the robot applies this user-chosen style across each task it encounters.

*common styles can be captured by a* canonical space.

We leverage our insight to introduce **PECAN**: **Per**sonalizing Robot Behaviors through a Learned **Can**onical Space. At training time, this algorithm collects human demonstrations of diverse tasks (e.g., decelerating at a red light, yielding at an intersection). The robot then leverages our proposed representation learning approach to autonomously extract a high-level canonical space from the task demonstrations. This learned canonical space is an abstract but user-friendly manifold: each point in the manifold corresponds to a different underlying style. At run time, humans select a point from this canonical space (i.e., each user can choose their own desired style), and the robot decodes that point into consistent behaviors across each task in the dataset. For example, if the current passenger selects a point that corresponds to defensive driving, the autonomous car will apply this canonical style to each individual task: decelerating gradually at stop lights, staying far from other vehicles on the highway, and yielding the right of way at intersections.

Overall, PECAN enables humans to rapidly personalize the way a robot behaves by directly specifying their preferred style. We make the following contributions:

**Learning a Canonical Space of Styles.** We introduce an approach for learning a high-level representation of robot behaviors through weak supervision. Given data of diverse human demonstrations and a few labels for demonstrations with similar styles, the robot encodes information about the tasks and styles into separate spaces. This allows users to choose their preferred style independent of the tasks.

**Forming a User-Friendly Canonical Space.** We propose characteristics of the canonical space — such as consistency and monotonicity — that make it easy for users to understand how the styles are encoded and select their preferred style. The robot induces these characteristics by using demonstrations that represent the extremes of the style spectrum.

**Evaluating with Real Users.** We compare our approach to state-of-the-art baselines in two user studies: one using a robot arm and another focusing on an autonomous driving scenario. Our results suggest that participants find it easier to personalize robot behaviors using an interface that leverages our approach. They also prefer PECAN over the baselines, stating that it is more consistent and intuitive. These results become more pronounced as the users gain experience using PECAN, suggesting that familiarization is an important factor in our method's effectiveness.

## 2 RELATED WORK

**Preference Learning.** Humans can personalize a robot by providing feedback about its behaviors. This includes demonstrating the desired motion [22, 25], correcting the robot's actions [16, 17, 30], selecting their preferred trajectory from options presented by the robot [27, 32, 36], or some combination of these feedback types [3, 13, 20, 21]. These works parameterize the robot's behavior with user-defined features, such as the speed of an autonomous car or its distance from other cars in a driving scenario. The parameters or weights for these features are then estimated based on the human's feedback. Together, the weights and features determine the robot's *style* (i.e., how it performs the given task) [26].

Specifying the correct set of features for modeling complex styles can be challenging [6]. To overcome this, previous research has also proposed deep reinforcement learning from human feedback [9, 12], which bypasses the need to specify such features. However, this approach is often time-consuming, as users may have to provide feedback hundreds or even thousands of times throughout the learning process.

Our work is different from these approaches in two ways. First, we do not pre-define the styles or features. Instead, we learn a latent representation of the styles from offline human demonstrations. Second, we do not collect more data from users in the task space. Instead, we design a canonical space that allows users to select their preferred style by simply clicking on the learned representation.

**Multi-task Personalization.** The methods discussed so far involve a single robot model that learns both the task and the user's preferred way of performing it. However, our insight is that these style preferences are often shared across diverse tasks. For instance, humans tend to prefer similar navigation styles across different search and rescue scenarios [11]. Some recent methods account for this by separately learning the user styles from the task-specific objective [2, 7, 33]. However, these works still assume that the relevant features are known. Our approach will also separate the learning of user styles from the tasks, but without assuming any features.

Specifically, we propose to construct a *canonical* representation of styles that is shared across multiple tasks. Previous research in multi-task learning has similarly explored how robots can learn canonical representations over several tasks [1, 19, 24, 28]. For example, in [19, 24], the robot learns a common visual representation of various tasks, and in [1], the robot learns a latent action representation that applies to a family of tasks. Our approach differs from these works because — in addition to learning multiple tasks — we also learn a canonical space that captures the different styles with which these tasks can be performed.

**Learning a Canonical Style Space.** Most relevant to our approach are methods like [18, 23, 29, 34] that learn a canonical representation of robot skills or styles over several tasks.

In [18], the robot embeds action sequences into a latent space of task-agnostic skills. The robot then executes these skills on a continuous range of tasks that are specified by their start and end states. When the set of tasks is finite, [23] learns a latent space comprised of discrete and continuous portions. The discrete variables capture the tasks and the continuous variables capture the latent styles. Both these methods employ Variational Autoencoders (VAEs) to train the latent spaces in an unsupervised manner. However, [18] assumes that the tasks are specified by the user, while [23] does not guarantee that the respective latent spaces will exclusively capture the tasks and styles present in the data.

To address this, [34] utilizes a small set of labels for trajectories that belong to the same task and for trajectories that correspond to the same skill. These labels are used to train Gated VAEs [31] which encode the task and skill knowledge into separate latent spaces. In contrast, [29] encodes classes (i.e., tasks) and styles as a Gaussian mixture within a single continuous latent space. Each Gaussian represents a class and its variance captures the styles. As opposed to [34], this method only requires the task labels for a subset of the inputs.

A major limitation of all these methods is that the encoded styles are not consistent across tasks. The same latent value can produce different styles depending on the task — making the interface unintuitive for humans. Ideally, users should be able to select their preferred style with a single click and produce similar robot behaviors across all tasks.

Similar to approaches like [23], we utilize separate discrete and continuous latent spaces to encode the task and style information. The latent style space becomes our canonical space. To ensure that the canonical space is consistent across tasks, we use a small set of labels for trajectories having similar styles but in different tasks. Unlike [34] and [29], our approach does not require any task labels. Instead, we capture the actual tasks and styles with their respective latent spaces by only using labels for trajectories with similar styles.

## 3 PROBLEM STATEMENT

We explore how robotic systems (such as robot arms or autonomous vehicles) can learn a canonical space for personalizing their behaviors. We assume that the robot is given a dataset with demonstrations of diverse tasks and ways of performing those tasks. From this dataset the robot needs to extract a low-dimensional and user-friendly manifold that embeds the *styles* shared across tasks (e.g., driving an autonomous car defensively or aggressively). Importantly, we do not assume that the styles are predefined or that the tasks are known. Instead, the system must learn these underlying styles to autonomously construct the canonical space.

**Trajectories.** Let $s \in \mathcal{S}$ be the system state and let $a \in \mathcal{A}$ be a robot action. For example, in our driving scenario the state $s$ includes the position and heading of the autonomous car and any other nearby vehicles, and $a$ is the robot's steering and acceleration inputs. A trajectory $\xi \in \Xi$ is a sequence of $T$ state-action pairs: $\xi = \{(s_1, a_1), \dots (s_T, a_T)\}$. We obtain trajectories by rolling out the robot's learned behaviors in the environment, or by collecting demonstrations from humans.

**Dataset.** At training time the robot is given a dataset with $N$ demonstrations from one or more human teachers. Each demonstration is a trajectory, so that the dataset consists of: $\mathcal{D} = \{\xi_1, \dots, \xi_N\}$. The trajectories within $\mathcal{D}$ show examples of multiple *tasks*, and perform those tasks with a variety of different *styles*. Let $\tau \in \mathcal{T}$ be the space of tasks and let $\theta \in \mathbb{R}^{d_\theta}$ be the space of styles. We assume that there are a discrete set of tasks (e.g., slowing for a red light, passing on the highway, crossing an intersection), but the manifold of styles is continuous. Returning to our driving example, the human can provide trajectories that slow for a red light (i.e., the task) along a spectrum from very gradually to very abruptly (i.e., the style). Overall, each demonstration $\xi \in \mathcal{D}$ corresponds to some task $\tau$ and style $\theta$.

**Labels.** In practice, however, we *do not assume* that the robot knows the task or style for any trajectory $\xi \in \mathcal{D}$. This is partially because it is difficult for humans to quantify the style of their demonstrations [15]. Imagine a person showing an autonomous car how to smoothly slow down for a red light; what numerical value of $\theta$ should the human give to that behavior? Rather than asking humans to provide $\theta$, we instead ask users to label trajectories that have *similar styles*. For our driving example, perhaps in $\xi_1$ the autonomous car brakes late for a red light, and in $\xi_2$ the autonomous car tailgates directly behind another vehicle. A human teacher might label $\xi_1$ and $\xi_2$ as having similar styles, since in both trajectories the robot drives aggressively. More generally, it

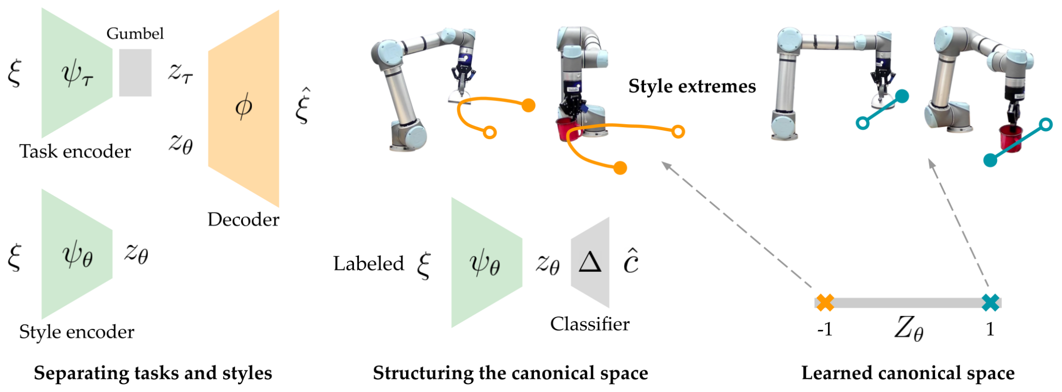

Fig. 2. Proposed architecture for **Pe**rsonalizing Robot Behaviors through a Learned **Can**onical Space (PECAN). (Left) The robot uses a task encoder $\psi_\tau$ and a style encoder $\psi_\theta$ to embed input demonstrations $\xi \in \mathcal{D}$ into two low-dimensional spaces: a latent task space $Z_\tau$ and a latent style space $Z_\theta$. A decoder network $\phi$ takes the combined latent tasks and styles as input and reproduces the input demonstrations. For labeled demonstrations, a classifier network $\Delta$ predicts the class labels from their latent styles. We train both the encoders and the decoder to accurately reconstruct the demonstrations. Simultaneously, we also train the style encoder along with the classifier such that it assigns similar latent values to trajectories with the same label. (Right) We show that when labeled demonstrations represent the extreme ends of the style spectrum, the canonical space is organized so that the latent styles of these extremes are positioned at the corners. This arrangement allows users to interpolate between the extremes by choosing intermediate latent values.

is up to the human teacher(s) to decide what the styles are, and what groups of trajectories they think have similar styles. As a result of this process the robot is given labels $\mathcal{Y}$. Each label $y_i \in \mathcal{Y}$ contains a set of trajectories $y_i = \{\xi_1, \xi_2, \ldots\}$ that all have similar styles (as determined by the human teachers). Every trajectory $\xi \in y$ belongs to the dataset $\mathcal{D}$; however, not all trajectories $\xi \in \mathcal{D}$ need to be labeled in $\mathcal{Y}$.

Overall, the robot's objective is to leverage the dataset $\mathcal{D}$ and labels $\mathcal{Y}$ to learn a canonical space of styles that allows users to easily personalize the robot's behavior across tasks.

## 4 LEARNING A CANONICAL STYLE SPACE

We want to enable people to select their preferred style $\theta$ for completing a collection of tasks $\mathcal{T}$ with just a few clicks. This personalization is challenging because the robot has no explicit knowledge of either the tasks or styles. However — recalling our motivating hypothesis — we recognize that styles are often shared across tasks, and so we can try to capture these underlying styles with a learned *canonical space*.

In this section we outline our approach for constructing this canonical space (see Figure 2). First, in Section 4.1, we introduce an autoencoder architecture that extracts the task and style information from the demonstrated trajectories. Our architecture encodes the tasks and styles into distinct latent spaces; the latent style space becomes our canonical manifold. Next, in Section 4.2, we propose characteristics that make the canonical space user-friendly, so that humans can easily interact with that space to search for and select their preferred styles. Finally, in Section 4.3, we describe our training process, focusing on how we learn a canonical space that effectively captures the styles in dataset $\mathcal{D}$ while also exhibiting user-friendly characteristics.

### 4.1 Separately Encoding Tasks and Styles

As defined in Section 3, each robot trajectory corresponds to a specific task $\tau \in \mathcal{T}$ and style $\theta \in \mathbb{R}^{d_\theta}$. For example, a robot arm could perform tasks like placing a cup in front of the user or pouring coffee into that cup. Different users may prefer different styles for these tasks (see Figure 2): perhaps one user provides demonstrations where the robot follows the shortest path, while another user shows demonstrations that take an exaggerated path to maintain a safe distance from the human.

Our insight is that these underlying styles are often consistent across tasks. We therefore want to learn a style space that is *independent* of the tasks, so that users can select their preferred style from this space and obtain the corresponding robot trajectory across each task. To facilitate this, we propose an autoencoder architecture with two encoders: a *task encoder* $\psi_\tau$ and a *style encoder* $\psi_\theta$, as well as one *trajectory decoder* $\phi$.

**Task Encoder.** The task encoder maps input trajectories $\xi \in \Xi$ to a latent space of tasks $Z_\tau$.

$$\psi_\tau : \Xi \mapsto Z_\tau$$

This latent space encodes the tasks present in our dataset. We assume that data consists of a finite number of discrete tasks $\mathcal{T}$. To capture these tasks, we want the latent task space to also be discrete such that each $z_\tau \in Z_\tau$ corresponds to a task $\tau \in \mathcal{T}$. Therefore, we discretize the output of the task encoder by applying the Straight-Through Gumbel Estimator [14]. This technique constrains $z_\tau$ to be a one-hot vector of dimensions $d_\tau = |\mathcal{T}|$. For instance, in the example shown in Figure 2, the task of placing the cup could be mapped to $z_\tau = [0, 1]$ while the task of pouring coffee is mapped to $z_\tau = [1, 0]$. In our experiments we will assume that the number of tasks $|\mathcal{T}|$ is known, but unsupervised metrics such as Normalised Mutual Information (NMI) can also be used to autonomously estimate the number of discrete tasks in the dataset [38].

**Style Encoder.** The style encoder $\psi_\theta$ maps input trajectories $\xi \in \Xi$ to a latent space of styles $Z_\theta$. We will refer to this latent style space as our *canonical space*:

$$\psi_\theta : \Xi \mapsto Z_\theta$$

Unlike the discrete space of tasks, we recognize that the robot styles can be continuous. For example, the robot trajectory in Figure 2 can vary along a continuous spectrum from straight goal-directed paths to exaggerated motions that stay far from the human. Accordingly, our canonical space $Z_\theta$ is a continuous manifold. In our experiments we use a $Tanh(\cdot)$ activation function at the final layer to bound the canonical space within $[-1, 1]^{d_\theta}$, so that our resulting canonical space is a $d_\theta$-dimensional cube.

**Trajectory Decoder.** Our goal is to let users choose a latent style $z_\theta$ from the canonical space and have the robot perform trajectories aligned with that style in each task $z_\tau \in Z_\tau$. To achieve this, we include a decoder network $\phi$ that takes both the task and style encodings as input and reconstructs the corresponding robot trajectories:

$$\phi : Z_\tau \times Z_\theta \mapsto \Xi$$

We train the decoder to accurately reconstruct a trajectory $\xi$ given the values for its latent task $z_\tau$ and latent style $z_\theta$ by minimizing the following loss:

$$\mathcal{L}_{trajectory} = \sum_{\xi \in \mathcal{D}} || \, \xi - \phi(\psi_\tau(\xi), \, \psi_\theta(\xi)) \, ||^2 \tag{1}$$

When this loss is minimized, it indirectly encourages the task and style encoders to capture sufficient representations of the trajectories in the dataset. However, this still does not guarantee that the representation captured in $Z\tau$ aligns with the actual tasks $\mathcal{T}$, or — along the same lines — that the representation in $Z_\theta$ aligns with the styles $\theta$.

To address this, we propose using a small set of labels $\mathcal{Y}$ for trajectories in the dataset. These labels identify trajectories from *different tasks* that share *similar styles*. In Section 4.3, we will introduce an additional loss function that leverages these labels to ensure that the representation captured in $Z_\theta$ aligns with the styles $\theta$. Further, we will show that by minimizing this loss together with $\mathcal{L}_{trajectory}$, we can also ensure that the latent tasks $z_\tau$ align with the actual tasks $\tau$.

## 4.2 Characteristics of a User-Friendly Canonical Space

So far we have discussed how our approach can accurately reconstruct robot trajectories using latent representations of the tasks and styles. However, merely learning latent spaces that can map to the correct behavior is not sufficient, since — by themselves — these latent spaces may not be easy for humans to interact with. Our goal is to learn a canonical space where the human can intuitively click around the manifold to specify their desired style. For instance, given a range of values from $-1$ to $+1$ as shown in Figure 2, how will the user know which regions of the canonical space correspond to straight or exaggerated trajectories? We now propose ways to structure the robot's learning so that the resulting canonical space is more user-friendly.

One way users can build an understanding of the canonical space is by selecting $z_\theta$ values and then visualizing the resulting robot trajectories across tasks $z_\tau$. For example, in Figure 2 the human might click on $z_\theta = +1$ in the learned canonical space, and then observe how the robot arm moves in a straight line to its goal. But for this interactive approach to be effective, the canonical space needs to be organized such that users can quickly find their desired style by visualizing as few $z_\theta$ values as possible. We therefore propose structuring the canonical style space to have the following user-friendly characteristics:

- *Consistency.* The latent values should result in consistent robot styles $\theta$ across tasks. For example, if $z_\theta = +1$ corresponds to a straight line path for the task of placing a cup, the same $z_\theta$ should also result in a straight line path for the task of pouring coffee. Therefore, for any $z_\theta \in Z_\theta$:
$$\theta(\phi(z_\tau, z_\theta)) \approx \theta(\phi(z'_\tau, z_\theta)) \quad \forall\, z_\tau, z'_\tau \in Z_\tau \tag{2}$$
This will allow users to customize the robot's style across all tasks by setting the desired latent style just once.
- *Monotonicity.* The styles should vary monotonically as we move from one point in the latent space to another. For example, decreasing the latent style value from $z_\theta = +1$ to $z_\theta = -1$ should gradually change the robot's style from straight-line paths to increasingly exaggerated arm motions. Therefore, for all $z_\theta, z'_\theta \in Z_\theta$:
$$(z_\theta - z'_\theta)(\theta(\phi(z_\tau, z_\theta)) - \theta(\phi(z_\tau, z'_\theta))) \geq 0 \tag{3}$$
This will enable users to easily find their desired style by interpolating between the extreme ends of the canonical space.

## 4.3 Semi-supervised Learning

We now present our complete training process for learning latent representations of the actual tasks and styles in our dataset, and inducing the user-friendly characteristics — *consistency* and *monotonicity* — in the learned canonical space.

**Labeling Style Extremes.** We first obtain a small set of labels $\mathcal{Y}$ for trajectories having similar styles but from different tasks. Specifically, we only obtain labels for trajectories that represent the *extremes* of the style range, e.g., the most or least exaggerated arm motions in Figure 2. We believe that labeling the extremes is easier than labeling intermediate styles. For instance, users find it difficult to distinguish between slightly different arm motions [4]. Note that we do not ask users to specify the actual style of a trajectory. We only assume that trajectories with the same label have similar styles, and correspond to one of the extremes of the canonical space.

**Style Classifier.** Minimizing the loss $\mathcal{L}_{trajectory}$ introduced in Section 4.1 trains the decoder to accurately reconstruct trajectories. Here we include an additional loss to ensure that the canonical space captures the actual styles and is *consistent* across tasks and *monotonic* along each axis.

We consider each $y_i \in \mathcal{Y}$ to be a separate class with one-hot labels $c(y_i)$, where $\mathcal{Y}$ contains $m$ classes. At training time, we pass the labeled trajectories $\xi_j \in y_i$ through the style encoder $\psi_\theta$ to obtain their latent styles $z_{\theta,j}$. We then feed the latent styles into a classifier network $\Delta$ that maps each latent value to a $m$-dimensional vector of class probabilities $p_j = [p_1, \dots, p_m]$ such that $\sum_{k=1}^{m} p_{j,k} = 1$ for any $p_j \in \mathcal{P}$.

$$\Delta : Z_\theta \mapsto \mathcal{P}$$

The classifier consists of a single fully-connected layer followed by a softmax layer. We train the style encoder and classifier to predict the class labels by minimizing the cross-entropy loss:

$$\mathcal{L}_{ce} = - \sum_{y_i \in \mathcal{Y}} \sum_{\xi_j \in y_i} \sum_{k=1}^{m} c_k(y_i) \log(p_{j,k}) \tag{4}$$

The subscript $k$ refers to the $k$-th index in the $m$-dimensional vectors of class labels and their probabilities. Intuitively, this loss encourages trajectories with the same style label $y$ to encode to nearby values within the canonical space, and trajectories with different labels to map to values far from one other in the canonical space. Particularly, since we only obtain labels for trajectories that represent the extremes of the styles spectrum, the latent style for each extreme will be placed in opposite corners of the canonical space.

We apply the following theorem from prior work [10] to show that when the $\mathcal{L}_{ce}$ loss is minimized, trajectories with the same label are encoded to the same latent value and the value for each label is positioned in a different corner of the latent space.

**Theorem.** Consider a latent space $Z = \{z \in \mathbb{R}^d : ||z|| \leq \rho_Z\}$ with a radius of $\rho_Z > 0$ and a linear classifier with weights $W \in \mathbb{R}^{m \times d}$. Given $N$ latent values $Z = \{z_1, \dots, z_N\}$ from this space and a balanced set of labels $Y$, the cross-entropy loss $\mathcal{L}_{ce}$ is bounded as:

$$\mathcal{L}_{ce}(Z, W; Y) \geq \log\left(1 + (m-1)\exp\left(-\rho_Z \frac{\sqrt{m}}{m-1} ||W||_F\right)\right) \tag{5}$$

This bound is tight if there are points $\zeta_1, \dots, \zeta_m \in \mathbb{R}^d$ that satisfy the following conditions [10]:

(1) $\forall n \in [N] : z_n = \zeta_{y_n}$

(2) $\{\zeta_y\}_y$ form a $\rho_Z$-sphere-inscribed regular simplex

(3) $\exists \rho_W > 0 : \forall y \in \mathcal{Y} : w_y = \frac{\rho_W}{\rho_Z} \zeta_y$

Condition (1) states that loss $\mathcal{L}_{ce}$ is minimized when latent values $z_n$ with the same label $y_n$ converge to a common point $\zeta_{y_n}$ in the latent space. This means that trajectories with the same style label will be encoded to the same latent style value even if they belong to different tasks.

Conditions (2) and (3) state that the points $\zeta_1, \dots, \zeta_m$ and weights corresponding to each class must inscribe a regular simplex in the latent space. This means that the latent values will be positioned at the edges of the latent space, with the points for different labels being equally distant from one another. For instance, consider the 1D canonical space illustrated in Fig. 2. If we have labels for $m = 2$ classes representing the style extremes, the latent values for the labeled trajectories will converge to two distinct points $(\zeta_1, \zeta_2)$ — one for the most exaggerated trajectories and one for the least. A regular simplex inscribed in this canonical space would be a line between end-points

$\zeta_1 = -1$ and $\zeta_2 = +1$. Therefore, when $\mathcal{L}_{ce}$ is minimized, the latent values for the style extremes will be pushed to the opposite ends of the canonical space.

We take advantage of these conditions to learn a canonical space that captures the actual styles and exhibits the desired user-friendly characteristics as follows:

**Combined Loss.** We simultaneously train all networks by minimizing the combined loss $\mathcal{L}$.

$$\mathcal{L} = \mathcal{L}_{trajectory} + \mathcal{L}_{ce} \tag{6}$$

First, we examine how minimizing $\mathcal{L}$ allows us to represent the actual tasks and styles in the data using their respective latent spaces. According to condition (1), minimizing the $\mathcal{L}_{ce}$ loss causes all trajectories with the same label to be encoded to the same latent value. For example, consider a label with two trajectories, $y_i = \{\xi_1, \xi_2\}$. When $\mathcal{L}_{ce}$ is minimized, the style encoder $\psi_\theta$ will map both trajectories to the same latent style, i.e., $\psi_\theta(\xi_1) = \psi_\theta(\xi_2) = z_{\theta,i}$. To simultaneously minimize $\mathcal{L}_{trajectory}$, the decoder $\phi$ must reconstruct the trajectories from this same latent style value. Recall that we assume each label has trajectories with similar styles but from different tasks, meaning $\xi_1 \neq \xi_2$. To output two different trajectories given the same latent style as input, i.e., $\phi(\psi_\tau(\xi_1), z_{\theta,i}) \neq \phi(\psi_\tau(\xi_2), z_{\theta,i})$, the decoder will require the trajectories to have different latent task values. Therefore, the task encoder must learn to map these trajectories to distinct values in the latent task space, i.e., $\psi_\tau(\xi_1) \neq \psi_\tau(\xi_2)$. In this way, we can train the task and style encoders to embed the actual tasks and styles in their respective latent spaces.

Next, we explore how condition (1) enables the style encoder to construct a *consistent* canonical space. Following the previous example, we see that minimizing $\mathcal{L}_{ce}$ trains the style encoder to map trajectories from different tasks to the same latent value. Since these trajectories belong to the same label, they have the same actual style. This results in a *consistent* canonical space where a given latent value corresponds to trajectories with similar styles across different tasks.

Lastly, according to condition (2), minimizing $\mathcal{L}_{ce}$ places the latent values for the extreme styles at the opposite ends of the canonical space. In practice, we find that training the style encoder to minimize both $\mathcal{L}_{trajectory}$ and $\mathcal{L}_{ce}$ causes the latent values of trajectories with intermediate styles to be placed *monotonically* between the extremes.

**Summary.** At training time, our architecture leverages a dataset of user demonstrations $\mathcal{D}$ and a small set of labels $\mathcal{Y}$ to learn a latent task space $Z_\tau$ and a canonical space of styles $Z_\theta$. We structure the canonical space to be consistent and monotonic by optimizing the combined loss in Equation (6). At runtime, the user selects a latent style $z_\theta$ from the learned canonical space. The decoder $\phi$ then takes this latent style as input and reconstructs the corresponding robot trajectory for each $z_\tau \in Z_\tau$.

In the following sections we will experimentally demonstrate the ability of our proposed architecture to learn distinct latent spaces for tasks and styles. We will also evaluate the accuracy of the trajectories generated from these latent representations, and assess whether the learned canonical space maintains our desired, user-friendly characteristics.

## 5 SIMULATION EXPERIMENTS

Here we perform controlled simulations to analyze the contributions of each component of PECAN. We compare the performance of our proposed approach to a state-of-the-art baseline for learning latent style representations [29] and ablations of our method in two environments: autonomous driving and robot manipulation (see Figure 3).

**Environments.** In the first environment we personalize the driving style of an autonomous car across two tasks, **Highway** and **Intersection**. In *Highway* the autonomous car follows another car on a highway. In *Intersection* the autonomous car waits for another car to pass before safely crossing an intersection. In both tasks, we consider 2D styles $\theta = [\theta_1, \theta_2]$ that define how aggressively or defensively the car drives. Here $\theta_1$ corresponds to the maximum speed the car achieves in an empty section of the road, and $\theta_2$ represents the minimum distance that the car keeps from other vehicles on the road. For example, some users may prefer a high speed of $\theta_1 = 100$ km/h until their car is within $\theta_2 = 30$ feet of the next car. Other users may prefer a slow speed of $\theta_1 = 40$ km/h but get as close as $\theta_2 = 10$ feet of the next car. We implement these tasks using the CARLO simulator [8].

In the robot environment we move away from the conventional meaning of tasks and styles to showcase the versatility of our approach. We consider three different robot platforms as the tasks: a **Kuka**, **Panda**, and **UR5**. In each task (i.e., for each type of robot) the goal is to transfer a cereal box from one bin to another. The styles are three dimensional $\theta = [\theta_1, \theta_2, \theta_3]$, and represent variations in environment. The variable $\theta_1$ is the orientation of the cereal box, while $\theta_2$ and $\theta_3$ mark the position of the target bin along the $x$ and $z$ axis respectively. We implement this environment using Robosuite [37].

**Baselines.** We compare **PECAN** to the following methods:

- **Ours-L**: An ablation of our approach that does not use any labeled data and only trains using the $\mathcal{L}_{trajectory}$ loss. Since it does use labels for trajectories with similar styles, we expect this approach to learn a canonical space that is not consistent across tasks, similar to prior work [23].
- **Ours-X**: An ablation of our approach that uses labels for trajectories with intermediate styles, instead of the style extremes. We expect such an approach to learn a canonical space that is consistent but not monotonous.
- **SeGMA**: A state-of-the-art approach for learning latent styles across multiple classes [29]. Instead of learning two separate task and style spaces, this method learns one combined latent space where the classes (i.e., tasks) are represented as Gaussians and their variance captures the styles. A latent style $z_s$ can be transferred from one task centered at $\mu_s$ to another task centered at $\mu_t$ by:

$$z_t = z_s + (\mu_t - \mu_s)$$

  This approach uses task labels instead of labels for trajectories with similar styles. Therefore we do not expect the latent styles to be consistent across tasks.

**Training.** In each environment we obtain a set of trajectories $\Xi$, where every trajectory $\xi \in \Xi$ corresponds to a different task and style $(\tau, \theta)$. In the driving environment, the trajectories are generated by simulated humans with different styles, while in the robot environment, the trajectories are teleoperated by an expert user. Next, we sample a small set of demonstrations $\mathcal{D}$ from the full set of trajectories $\Xi$ to train the methods. In the driving environment, we create a training dataset of 16 demonstrations from a set of 352 trajectories such that 8 demonstrations belong to *Highway* and 8 belong to *Intersection*. Four trajectories in each task correspond to the extreme styles — [high $\theta_1$, high $\theta_2$], [high $\theta_1$, low $\theta_2$], [low $\theta_1$, high $\theta_2$], and [low $\theta_1$, low $\theta_2$] — while the other four are randomly sampled. In the robot environment, we sample a training dataset of 27 demonstrations from a set of 60 trajectories. The data is balanced across the three tasks. Of the 9 demonstrations in a task, 8 represent the style extremes and 1 is randomly sampled.

For **Ours-X**, all trajectories in a task are randomly sampled. We train all approaches, except **Ours-L**, with the same amount of labeled data. **SeGMA** requires labels for trajectories that belong to the same task. In contrast, **Ours-X** and **PECAN** do not require any task labels and only use labels for trajectories with similar styles. Our code can be found here: https://github.com/VT-Collab/PECAN.

**Testing.** We test the performance of each method using the entire set of trajectories $\Xi$. Specifically, we measure the accuracy of encoding the tasks (*Task Accuracy*), the error in reconstructing the trajectories (*Trajectory Error*), and the *Consistency* and *Monotonicity* of the latent style space.

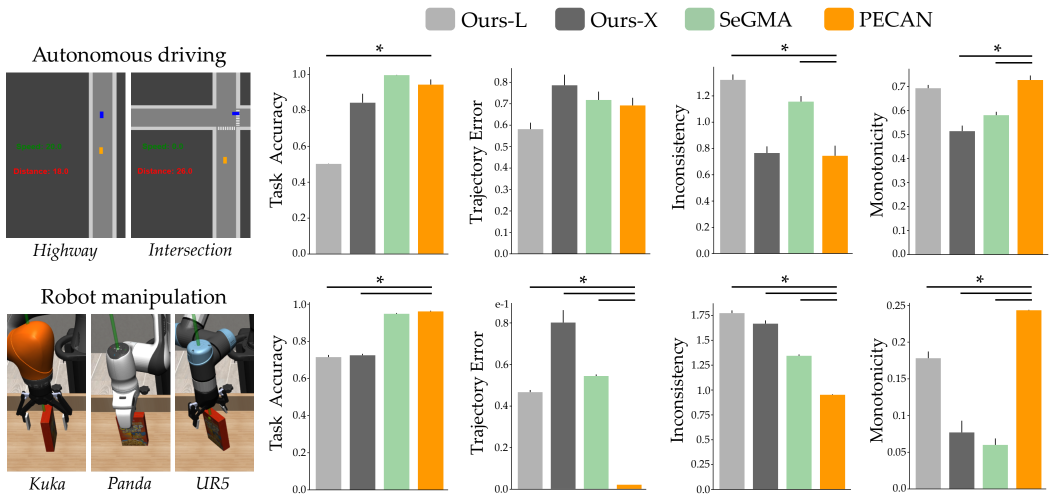

Fig. 3. Simulation results for autonomous driving (Top row) and robot manipulation (Bottom row) environments. We compare our proposed approach, PECAN, to a state-of-the-art baseline, SeGMA, and ablations of our approach, Ours-L and Ours-X. While SeGMA uses task labels, PECAN uses labels for trajectories with similar styles (specifically the style extremes). Both ablations use the same architecture as PECAN, however, Ours-L does not train with any labels, whereas Ours-X uses labels for trajectories with intermediate styles (instead of the style extremes). In both environments, PECAN achieves comparable *Task Accuracy* to SeGMA. Although PECAN has significantly lower Trajectory Error in the robot environment, its performance is similar to the baselines in the driving environment. The main advantage of PECAN over the baseline methods is that the canonical spaces learned by our approach are more consistent and monotonic (i.e., more user friendly). An asterisk (*) denotes statistical significance.

*Task Accuracy* is 1 if trajectories that belong to the same task are encoded to the same value in the latent task space, with distinct latent values for trajectories in different tasks. If all trajectories are mapped to the same latent task, the *Task Accuracy* is $1/|\mathcal{T}|$. *Trajectory Error* is the mean squared error between the original trajectory and the trajectory reconstructed from the latent style space. Next, as a proxy for measuring the *Consistency* of the canonical space, we measure its *Inconsistency* by computing the difference between the latent style values of trajectories that have similar styles but in different tasks. Inconsistency is defined as:

$$\mathbb{E}_{\xi_1,\xi_2\in\Xi}\ ||\psi_\theta(\xi_1)-\psi_\theta(\xi_2)|| \quad \text{if} \ \ \theta(\xi_1)=\theta(\xi_2) \ \ \text{and} \ \ \tau(\xi_1)\neq\tau(\xi_2) \tag{7}$$

Here we do not compute *Consistency* directly using Equation (2) because it requires access to a function $\theta(\cdot)$ that maps reconstructed trajectories to their actual style values. In our simulations, we only know the actual tasks and styles for the trajectories in the dataset $\Xi$, and not for their reconstructions. Lastly, we measure the *Monotonicity* of the canonical spaces by computing the correlation between the latent values and the actual styles of trajectories in $\Xi$. In a monotonous space, the difference in latent values of trajectories should be correlated to the difference in their styles. We measure this using Spearman's rank correlation coefficient [35]. A coefficient of 1 or $-1$ indicates perfect correlation, while 0 means that the styles and their latent values are uncorrelated. We take the absolute value of this coefficient as the *Monotonicity* of the canonical space.

**Results.** Our results are displayed in Figure 3. We calculated these results over 20 training runs, each starting with randomly initialized network weights. We performed one-way ANOVA tests and found that the choice of method had a significant effect on *Task Accuracy* in the autonomous driving ($F(3, 76) = 58.7$, $p < 0.01$) and robot manipulation ($F(3, 76) = 313.1$, $p < 0.01$) environments. In both environments, **SeGMA** achieved a high *Task Accuracy*, while **Ours-L** achieved the lowest. This is likely because **SeGMA** is trained with labels for the tasks, whereas **Ours-L** operates without any labels. **PECAN**, on the other hand, does not use task labels like **SeGMA**. Yet it achieved a comparable *Task Accuracy* by leveraging labels for trajectories with similar styles. Post-hoc comparisons indicated a statistically significant difference ($p < 0.01$) between the *Task Accuracy* of **PECAN** and **Ours-L** in both environments. On the other hand, there was no significant difference in the *Task Accuracy* of **PECAN** and **SeGMA** in the autonomous driving ($p = 0.57$) and robot manipulation ($p = 0.64$) environments.

**Our-X** also attains high *Task Accuracy* in the driving environment by leveraging the style labels similarly to **PECAN** ($p = 0.07$). However, in the robot environment, **Ours-X** has a significantly lower *Task Accuracy* ($p < 0.01$) due to a high error in its reconstructed trajectories. These findings demonstrate that to learn the correct task representation by only using labels for trajectories with similar styles, it is crucial to optimize both the reconstruction loss and the cross-entropy loss, as we theoretically suggested in Section 4.3.

Next, we observed that the canonical spaces learned by **PECAN** are more consistent and monotonous than those learned by any of the baselines. One-way ANOVA tests revealed that the choice of method had a significant effect on the *Consistency* ($F(3, 76) = 27.5$, $p < 0.01$) and *Monotonicity* ($F(3, 76) = 30.3$, $p < 0.01$) of the canonical space in the driving environment. Post-hoc comparisons indicated that the spaces learned by **Ours-L** show comparable *Monotonicity* ($p = 0.53$) to **PECAN** but lack *Consistency* ($p < 0.01$) because of not using any labels. In contrast, **Ours-X** manages to learn a consistent latent space by using the style labels similar to **PECAN** ($p = 0.99$) in the driving environment. However, it lacks *Monotonicity* ($p < 0.01$) because it obtains labels for the intermediate styles rather than the style extremes. **SeGMA** does not leverage any style labels and thus has a lower consistency ($p < 0.01$) and monotonicity ($p < 0.01$) than **PECAN**.

**Takeaways.** These results demonstrate that PECAN can successfully learn the task and style encodings by only using labels for trajectories with similar styles. While SeGMA achieves similar accuracy in encoding the tasks and reconstructing trajectories by using task labels, unlike PECAN, it does not learn a consistent and monotonic canonical space. We hypothesize that these characteristics make the canonical space intuitive for users to understand, enabling them to easily find a latent value corresponding to their preferred style.

Our ablations highlight that each component of PECAN is critical for constructing a user-friendly canonical space. The style labels ensure that PECAN learns a consistent canonical space and obtaining these labels for trajectories with extreme styles helps in making the space monotonic. Conversely, the canonical space learned by Ours-L is inconsistent because it does not use any labels, while the canonical space learned by Ours-X is consistent but not monotonic because it uses labels for intermediate styles as opposed to the style extremes.

## 6 USER STUDY

Our simulated experiments indicate that our proposed approach learns a consistent and monotonic space of styles. In this section, we will investigate whether these characteristics actually make the canonical spaces user-friendly, and whether users are able to leverage these spaces to directly personalize robot behaviors.

We conducted two in-person user studies to evaluate the effectiveness of PECAN with real users. In the first study, we compare two direct approaches for personalizing the trajectory of a robot arm using learned canonical spaces. Specifically, we test whether the consistent and monotonic

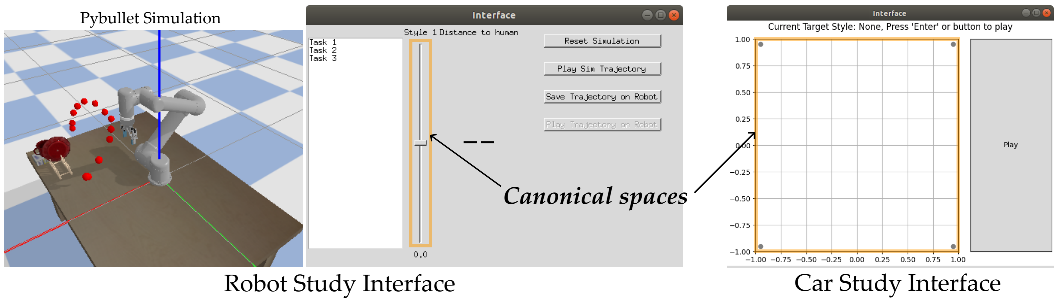

Fig. 4. Interfaces for personalizing the behavior of the robot (Left) and the autonomous car (Right) in our user studies. In the robot study, the canonical space was a 1D line which represented the distance that the robot maintains from the user. Users personalized the style of the robot's trajectory by moving the slider along the line. For tasks 1 and 2, users could visualize the robot's trajectory in a Pybullet simulation before executing it on the robot in the real world. In the car study, the canonical space was a 2D square which captured the maximum speed of the autonomous car and the minimum distance it maintains from other cars on the road. Users personalized the car's driving style by selecting different points inside the square. Since the driving environment was entirely in simulation, there was no need to visualize the car's trajectory separately before execution. Figure 3 shows examples of the simulated car in the Highway and Intersection tasks.

spaces learned by PECAN are more *intuitive* to users than the spaces learned by the state-of-the-art baseline, SeGMA. In the second study, we address the overarching question of how to best personalize robot styles: through *direct* selection in a style space, or via *indirect* methods that learn from user feedback. We compare our direct approach, PECAN, with a standard indirect method from prior work [27]. We test these approaches in the context of customizing the driving style of an autonomous car and assess the pros and cons of each method.

## 6.1 Learning User-Friendly Canonical Spaces

In our first user study, we tested if structuring the canonical spaces to be consistent and monotonic makes them more user-friendly for personalizing robot styles. A user-friendly space should be intuitive and easy to use, enabling users to quickly find their desired style. We compared two approaches for learning a canonical space of robot styles: PECAN and SeGMA [29]. Our simulations in Section 5 showed that both approaches effectively learned latent representations of the tasks and styles from robot trajectories. However, the spaces learned by PECAN were more consistent and monotonic as compared to those learned by SeGMA. Therefore, we hypothesized that users would find PECAN to be more intuitive and easier to use than SeGMA for personalizing robot styles.

**Experimental Setup.** Participants in this study interacted with a 6-DoF UR5 robot arm in three manipulation tasks: (i) handing over a plate to the user (**Handover Plate**), (ii) placing a cup in front of the user (**Place Cup**), and (iii) pouring coffee into the cup (**Pour Coffee**). In each task, the robot's style was defined by the distance it maintained from the user. At one extreme, the robot could follow a straight-line path that comes very close to the user, while at the other extreme, it could take a curved path that stays as far from the user as possible. To personalize the robot's trajectory, each user interacted with an interface containing a canonical space of the robot's styles. The interface featured a task selection menu, a slider for choosing the latent style (as shown in Fig. 4-Left), and buttons to execute the trajectory corresponding to the chosen style in a Pybullet simulation and on the real robot. We pre-programmed the robot to pick up the objects for each task

Table 1. Survey questions (Likert scales with 7-option response format). We grouped questions into five scales and tested their reliability using Cronbach's $\alpha$. The reliability scores presented in the table are based on the responses recorded in the robot user study.

| Questionnaire item | Reliability |
|---|---|
| **Easy** | |
| - It was easy to personalize the robot trajectory using this interface. | 0.89 |
| - It was challenging to personalize the robot trajectory using this interface. | |
| **Intuitive** | |
| - I was able to understand how moving the slider changed the robot trajectory. | |
| - It was difficult to understand how the robot trajectory would change by moving the slider. | 0.94 |
| - The interface was intuitive to use for personalizing the robot trajectory. | |
| - I did not find the interface intuitive for personalizing the robot trajectory. | |
| **Accurate** | |
| - In the end, I was able to accurately personalize the robot trajectory. | 0.88 |
| - In the end, I was unable to personalize the robot trajectory accurately. | |
| **Easy (No Visuals)** | |
| - It was easy to personalize the robot trajectory in the third task based on the first two tasks. | 0.93 |
| - It was challenging to personalize the robot trajectory in the third task based on the first two tasks. | |
| **Prefer** | |
| - Overall, I would prefer to use this interface to personalize the robot trajectory. | 0.85 |
| - Overall, I would not like to use this interface to personalize the robot trajectory. | |

(i.e., plate, cup, or kettle) from their initial positions. The robot then performed the tasks according to the style selected by the user. See videos here: https://youtu.be/wRJpyr23PKI

Participants performed the three tasks in the order given above. In the first two tasks, *Handover Plate* and *Place Cup*, they could preview the robot's trajectory in the Pybullet simulation before executing it in the real world. However, they did not have this option in the third task of *Pour Coffee* and had to find their target style based on their experience of using the interface in the first two tasks. We did this to determine if users could build an accurate understanding of the canonical space, enabling them to effectively transfer it to new tasks. Therefore, we treated the first two tasks as familiarization tasks and evaluated the user performances in the *Pour Coffee* task. We anticipated that a consistent and monotonic canonical space would be easier for users to learn and apply across new tasks without the need to visualize the robot's behavior.

**Independent Variables.** We compared our proposed approach (**PECAN**) to a state-of-the-art baseline for generating latent style representations (**SeGMA**). To train these methods, we provided 3 demonstrations each in the *Handover Plate* and *Place Cup* tasks. Two of those demonstrations represented the extreme styles in that task — trajectories with the maximum and minimum distance to the user. Additionally, we asked each participant to provide 2 demonstrations in the *Pour Coffee* task, one for each of the extreme styles. In total, we trained PECAN and SeGMA using a dataset of 8 demonstrations with labels for the trajectories with extreme styles. For PECAN, we assigned the same label to trajectories with similar styles across the tasks. On the other hand, we provided the same label for trajectories in the same task when training SeGMA.

**Participants and Procedure.** We recruited 14 participants (3 female, average age 28 ± 5 years) from the Virginia Tech community. Participants gave informed written consent under IRB #23-1237. At the start of the experiment, users kinesthetically guided the robot arm to demonstrate trajectories for the extreme styles in the *Pour Coffee* task. We then trained both methods on the user demonstrations along with the six previously collected demonstrations. Users interacted with the robot in all three tasks — once with each method. They completed the tasks in a fixed order but the ordering of the methods was counterbalanced: half of the users started with PECAN and the other half started with SeGMA.

In the *Handover Plate* and *Place Cup* tasks, users personalized the robot's trajectory once to match distinct target styles. Then, in the *Pour Coffee* task, users personalized the robot's motion twice to achieve two additional target styles. We randomly sampled the four target styles for each user. For each style, users had 3 attempts to perform the task on the real robot while ensuring that its trajectory stayed within a small tolerance of the desired distance. To help users gauge the actual style of the robot's trajectory, we displayed its maximum distance from the user on the interface.

**Dependent Variables.** For each task we recorded the number of attempts that users took to achieve the target style on the real robot (**Real Attempts**). We also recorded the number of times users visualized the latent styles in simulation before executing them in the real world (**Sim Attempts**) for the *Handover Plate* and *Place Cup* tasks. A higher number of attempts indicated that it was difficult for users to identify their desired styles in the learned canonical space. Specifically, in the *Pour Coffee* task, where users did not have the option to simulate the styles, a higher number of attempts indicated that the interface was not intuitive and consistent across tasks. We also measured the error in the distance (**Style Error**) and final position (**Task Error**) of the trajectories executed by users in their last attempt for each target style.

After working with each interface users answered a 7-point Likert scale survey (see Table 1). This survey measured their subjective responses along five scales: the *Intuitiveness* of the interface, how *Easy* it was to personalize the robot's style with that interface, how easy it was to personalize the robot's style without visual information (*No Visuals*), the *Accuracy* of the reconstructed trajectories, and if they *Preferred* using that interface. Users also detailed their experience after using each interface in an open-ended response.

**Hypothesis.** We had the following hypotheses:

> **H1.** *Users will find interfaces that use PECAN to be easier and more intuitive than those that use SeGMA for personalizing the robot trajectory.*
>
> **H2.** *Users will subjectively prefer using canonical spaces learned by PECAN over those learned by SeGMA.*

**Results.** Our results are presented in Fig. 5. In the first two tasks users required a similar number of real attempts for both methods. This was because they could spend ample time refining their desired style in simulation before executing it on the real robot. Therefore, we tested our first hypothesis by comparing the performance of the two methods in the *Pour Coffee* task. A two-tailed paired t-test showed a significant difference in the number of real attempts ($p < 0.01$) with PECAN and SeGMA. This suggests that PECAN is more intuitive and consistent than SeGMA, making it easier for users to find their target style. Subjectively, users reported that it was easier to personalize the robot with PECAN than SeGMA, especially in the third task where they had no visual information ($p < 0.05$). Users also reported that they found PECAN to be more intuitive than the baseline. A two-tailed paired t-test showed a significant difference in the combined ratings of the *Easy* ($p < 0.05$) and *Intuitive* ($p < 0.05$) scales for PECAN and SeGMA. This result supported hypothesis **H1**.

In total, 11 out of 14 users stated that they preferred working with interfaces trained using PECAN, giving it a significantly higher rating than SeGMA on the *Prefer* scale ($p < 0.05$). This result supported **H2**.

**Takeaways.** Overall, these results demonstrate that the consistent and monotonous spaces learned by our approach are more intuitive for users, making it easy for them to personalize the robot, especially when they cannot simulate the robot's motion before executing it in the real world. In

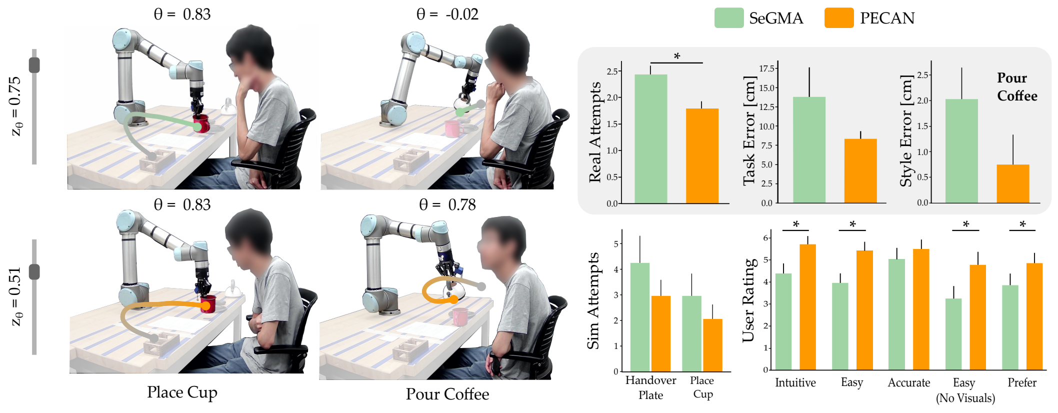

Fig. 5. Objective and subjective results from the robot user study. (Left) User applies the same latent value $z_\theta$ from *Place Cup* to *Pour Coffee* expecting a similar style across both tasks. PECAN produces trajectories with similar distances $\theta$ for *Place Cup* and *Pour Coffee*, but SeGMA generates a straight line trajectory for *Pour Coffee*. (Top Right) When using PECAN participants had lower *task error*, *style error*, and fewer *real attempts* ($t(13) = -3.12$, $p < 0.01$) performing new tasks without visual information. (Bottom Right) Subjectively participants *prefer* ($t(13) = 2.55$, $p < 0.05$) working with PECAN compared to SeGMA, as they found PECAN more *intuitive* ($t(13) = 3.35$, $p < 0.01$), and *easy* ($t(13) = 2.68$, $p < 0.05$) especially without visual information ($t(13) = 2.76$, $p < 0.05$). An asterisk (*) denotes statistical significance.

their open-ended response, five users stated that the slider (i.e., canonical space) for the baseline approach, SeGMA, was inconsistent across tasks. For example, one user wrote that "*this one [SeGMA] appeared to change for each task, which made it hard to get to understand the slider*". Therefore, users needed more attempts to achieve their target style in the *Pour Coffee* task with SeGMA as compared to PECAN. We also observed that users achieved a lower task and style error in the *Pour Coffee* task when using PECAN, although the difference was not statistically significant.

## 6.2 Direct vs. Indirect Personalization

In the second user study we determined the pros and cons of *directly* selecting the robot's style compared to *indirect* methods that estimate the style from user feedback (e.g., ranking robot trajectories). Participants were presented with two fundamentally different approaches for modifying an autonomous car's driving style — our direct approach, PECAN, and an established approach for indirectly learning from human preferences [27].

**Independent Variables.** Specifically, we compared **PECAN** to an active preference-based learning approach, which we refer to as **APReL**. We implemented this approach based on the code provided in [5]. At each step, APReL presented two trajectories to the users, each representing a different style, and asked them to choose their preferred trajectory. Based on their choices, APReL inferred the individual user styles. It strategically selected these trajectories to maximize the information it gained about the user's style from their choice. For example, showing trajectories with distinctly different speeds, such as one high-speed and one low-speed, is more informative than showing two trajectories with similar speeds. In our implementation, APReL selected the trajectories from a discrete space of 176 uniformly sampled styles. We trained PECAN with 24 demonstrations: 8 demonstrations corresponded to the extreme styles and 16 were sampled randomly, as explained in Section 5. It is important to note that, unlike our approach, APReL has direct access to the features that parameterize the driving style of the autonomous car. Therefore, we might expect this baseline to outperform our approach because it knows the actual styles, while our approach must learn these styles from user demonstrations.

**Experimental Setup.** The driving simulation in this study was the same as in Section 5. We had two tasks, **Highway** and **Intersection**, and 2D styles that depended on the speed of the autonomous car and the minimum distance that it maintains from other vehicles. For PECAN, users selected their preferred style by clicking on a point in a 2D canonical space as shown in Figure 4-Right. In contrast, APReL showed simulations of two car trajectories and asked users to select the trajectory that best matched their preferred style.

**Participants and Procedure.** We recruited 10 participants (2 female and 1 undisclosed, average age 27 ± 5 years) from the Virginia Tech community. None of these participants took part in our first user study. Participants gave informed written consent under IRB #23-1237.

We asked each user to personalize the driving style of the autonomous car to a randomly sampled style (i.e., car speed and following distance) across both tasks. Users customized the car's style using both direct (PECAN) and indirect (APReL) approaches. We counterbalanced the ordering of these approaches. When using PECAN, users clicked on points in the canonical space and visualized the car behavior until they found their target style. Importantly, we did not describe how the styles are distributed in this space. We only showed users the car's behavior for the latent values in each corner. For APReL, we explained that the interface will present two options and users must select the best option that trains the car to achieve their target style. After each selection, the interface updated its estimate of the user's style and showed the learned behavior. For both methods, users had to achieve the desired style within a tolerance of ±15 km/h speed and ±5 feet distance.

**Dependent Variables.** For the indirect approach (APReL) we recorded the total number of queries that users had to answer for personalizing the car's behavior. For the direct approach (PECAN) we counted the total number of points that users had to visualize for finding their preferred style. We collected the subjective responses of users for the same scales as in Table 1. We also added another scale for measuring if users perceived that they required fewer attempts (clicks or queries) to personalize the robot as they gained more experience with each method (*Learn*).

**Hypothesis.** We hypothesized that:

> **H3.** *Users will find it easier to personalize the driving style of the autonomous car with our direct approach (PECAN) as compared to the indirect baseline (APReL).*
>
> **H4.** *Users will prefer using our direct approach (PECAN) over the indirect baseline (APReL) for personalizing the driving style of the autonomous car.*

**Results.** Our results are summarized in Figure 6. Users were able to successfully personalize the style of the autonomous car with both direct (PECAN) and indirect (APReL) approaches. When using PECAN, 6 out of 10 users were able to personalize the car's style with a single click in at least one of the driving tasks! By contrast, there was only one instance when APReL learned the target style after a single query. Although users required fewer attempts (clicks or queries) on average to achieve their target style with PECAN ($M = 6.1$, $SE = 0.87$) than with APReL ($M = 7.2$, $SE = 1.05$), a two-tailed paired t-test did not show a significant difference ($t(9) = -1.14$, $p = 0.28$).

Overall, 7 out of 10 users stated in the survey that they preferred using our direct approach. While they gave a higher rating for PECAN ($M = 5.6$, $SE = 0.57$) than APReL ($M = 4.9$, $SE = 0.45$) on the *Prefer* scale, the difference was not statistically significant. We only saw a significant difference in their subjective ratings for *Learn* ($t(9) = 2.67$, $p < 0.05$).

**Discussion.** While a majority of the users performed slightly better with our direct approach (PECAN) and preferred it over the indirect approach (APReL), the differences were not sufficient to support either hypothesis. There are two potential reasons for this result:

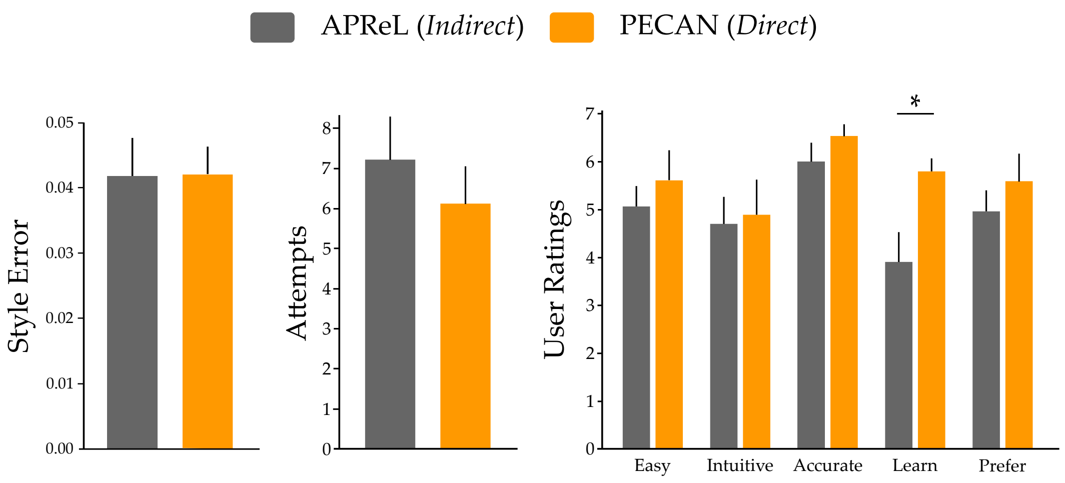

Fig. 6. Objective and subjective results for the second user study. (Left) The average error in the style of the personalized car trajectory in both tasks. (Center) The total number of attempts (queries or clicks) required by users to personalize the car in both tasks. (Right) Users found both the direct and indirect approaches to be *intuitive*, *accurate*, and *easy* to use. In particular, they perceived that they needed fewer clicks to find their desired style as they gained more experience with our direct approach.

First, users may have individual preferences for how they personalize the robot's behavior. For example, most users preferred PECAN because they liked that they could quickly change the car's behavior without waiting for it to learn, stating that it was "*quicker at learning and took fewer attempts*". On the other hand, some users preferred APReL since they found it more convenient to passively respond to queries than actively selecting their style, even if it took more time, stating that they liked to "*just observe and then decide on the preferred trajectory*".

Second, the baseline had direct access to the features that define the car's style. By contrast, our approach had to learn a representation of the styles from data. This meant that while the baseline could directly reason about the car's driving style, users had to spend some time with our interface to understand how the latent values mapped to actual styles. Notably, in this study, users did not have any practice time before using the interfaces. They only interacted with the interfaces 6-7 times on average. Based on the subjective responses of users for the *Learn* scale, we think that people can do better with more experience using PECAN. Hence, we conducted a follow-up study to further validate our findings and develop a better understanding of the advantages and disadvantages of using direct and indirect methods for personalizing robot behaviors.

### 6.3 Follow-up Study: Direct vs. Indirect Personalization

Participants in the second user study reported that they needed fewer attempts to personalize the car's behavior as they gained more experience with our direct approach. Therefore, in our follow-up study we compared our direct approach, PECAN, to the indirect baseline, APReL, with users who have practiced with both approaches.

We recruited 10 new participants (1 female, average age 20 ± 1 years) from the Virginia Tech community. None of these participants were involved in the first two studies. We followed the same procedure as the previous study with the following changes: (i) Before starting the experiment, we gave the participants 10 minutes with each approach to practice personalizing the driving style of the autonomous car. (ii) To obtain a more consistent measurement of their objective performance,

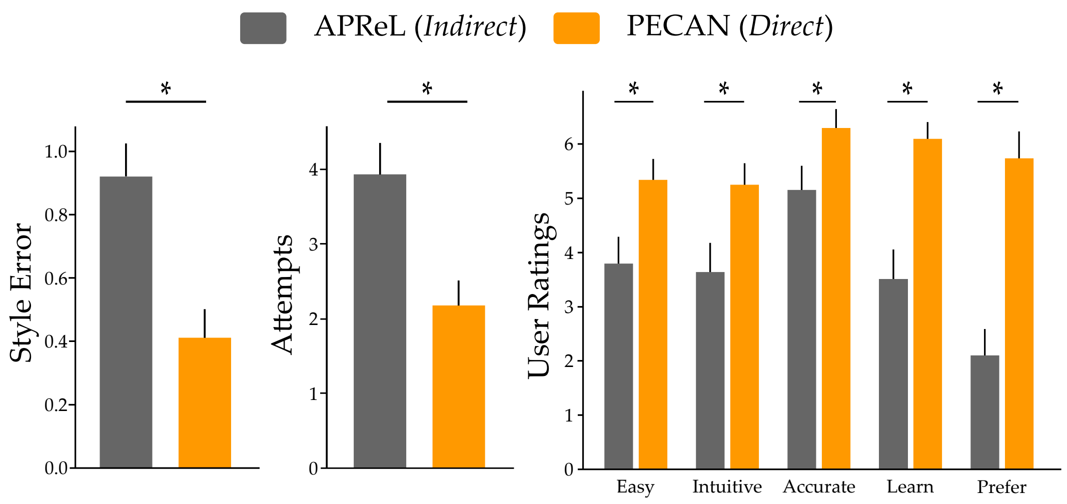

Fig. 7. Objective and subjective results for the follow-up study comparing direct and indirect approaches for personalizing robot styles with experienced users. (Left) The average error in the style of the personalized car trajectory in both tasks. (Center) The average number of attempts (queries or clicks) required by users to personalize the car for each target style across both tasks. After practicing with both approaches for 10 minutes, users were able to achieve their target style more accurately with our direct approach, PECAN, and needed fewer attempts to do so than the indirect baseline, APReL. (Right) Subjectively, experienced users found PECAN to be more *intuitive*, *accurate*, and *easy* to use than APReL. Consequently, they preferred it over the indirect baseline for personalizing the car's trajectory across tasks.

we asked the participants to personalize the car's trajectory for four distinct target styles, as compared to just one target style in Section 6.2. For each target, users were given a maximum of ten attempts (i.e., clicks or queries).

**Hypothesis.** We extended **H3** and **H4** to obtain the following hypotheses:

> **H5.** *After gaining experience with both approaches, users will find it easier to personalize the autonomous car's driving style with our direct approach (PECAN) than the indirect baseline (APReL).*
>
> **H6.** *After gaining experience with both approaches, users will prefer using our direct approach (PECAN) to the indirect baseline (APReL) for personalizing the car's style.*

**Results.** The results of the follow-up study are shown in Figure 7. After practicing with both approaches, we observed that users required significantly fewer clicks or queries to personalize the autonomous car's style with PECAN than with APReL. A two-tailed paired t-test showed a statistically significant difference ($t(9) = -3.28$, $p < 0.01$) between the average number of attempts required by users per style with PECAN ($M = 2.1$, $SE = 0.33$) and APReL ($M = 3.9$, $SE = 0.48$). Subjectively, experienced users found our direct approach, PECAN, to be significantly *easier* ($p < 0.01$) and *intuitive* ($p < 0.01$) than the indirect approach, APReL. This result supported our hypothesis **H5**.

Overall, 9 out of 10 users stated that they preferred to directly specify the style instead of providing indirect feedback, and gave a significantly higher rating ($t(9) = 3.98$, $p < 0.01$) for PECAN ($M = 5.7$, $SE = 0.48$) than APReL ($M = 2.1$, $SE = 0.48$) on the *Prefer* scale. This result supported our hypothesis **H6**.

Table 2. Open-ended responses from participants in the follow-up study that highlight the pros and cons of direct and indirect approaches for personalizing robot behaviors.

| Direct (PECAN) | Indirect (APReL) |
|---|---|
| ***Pros:*** | ***Pros:*** |
| "This interface felt more ***intuitive*** and made the experience feel more personable." | "I liked how the two ***options*** for trajectory that were given were ***different***." |
| "I liked the interface as I was able to get closer to the optimal point ***easily and quickly***." | "I thought that it was relatively easy to make changes." |
| "I liked that it was relatively easy to place points on the graph and see what reaction they had on the scenario." | |
| "Having an array of options rather than a binary decision made it feel ***much more personal*** and easy to use." | |
| ***Cons:*** | ***Cons:*** |
| "Definitely a bit tricky at first but once there were a few data points to base my entries off of it became much easier." | "It took many ***more tries*** for me to get the speed and distance close to the target." |
| "One thing I did not like about the interface was how the values were not on the grid." | "I overall did not like this as it seemed to give me substantially ***less control*** over what was happening." |
| "I did not like that the axes of the graph were **not labeled**." | "Sometimes I ***couldnt understand*** why the trajectory didnt change in the way I wanted." |
| "The one thing I struggled with is how the grid is ***not linear***." | "I did not like that I was unable to tune speed or distance independently." |

**Takeaways.** In this follow-up study users had the opportunity to practice personalizing the car's driving style with both direct (PECAN) and indirect (APReL) approaches. We did not describe how the styles were distributed in the canonical space for PECAN nor did we explain how APReL learned from the user's choices. After just 10 minutes of practice, users found PECAN to be easier and more intuitive than APReL, enabling them to achieve the target styles in fewer attempts.

Each user was tasked with personalizing the car's motion for four different target styles, allowing them to interact with each approach more times than in the previous study. Here we observed that in a few instances, APReL presented users with two trajectory options, neither of which aligned with their target style. For example, one user stated that "*often, my options were to increase my speed and distance or decrease the speed and distance when I needed to tune them opposite of each other*". Such instances were confusing for users, causing them to not achieve their target styles within ten attempts. Therefore, users had a significantly higher style error with APReL than with PECAN.

We summarize the feedback provided by users on the advantages and disadvantages of both approaches in Table 2. We have included all comments: we only omit redundant comments and surplus details for clarity. Overall, users found that directly personalizing the car's style using PECAN was more intuitive, easy, quick, and personable. Although understanding how the latent values mapped to actual styles was initially tricky, users reported that it became much easier after a few tries. Conversely, while users appreciated the options presented by the indirect approach (APReL), they felt they had less control over its learning process and sometimes struggled to understand how their choices affected the car's learned behavior.

A common critique of our direct approach was that users wanted the styles and axes to be labeled in the canonical space. However, unlike the indirect approach, PECAN does not know the actual styles. We only use weak supervision to learn the canonical space from demonstration data. To enumerate the actual styles in the canonical space, we would need labels that specify the exact styles of trajectories in the dataset, not just whether they have similar styles. Another critique was that, although our canonical space was monotonic, users would have preferred it to be linear. This meant that, after visualizing the car's style for a couple of points in the canonical space, users could estimate the direction in which their target style would lie in the space, but not the exact distance. We aim to address this issue in future work by inducing proportionality in our canonical space. Despite this limitation, our results show that a monotonic canonical space is sufficient for users to find their desired style in a few clicks.

## 7 CONCLUSION

In this paper we enabled humans to directly personalize robot behaviors through a canonical style space. We first introduced PECAN, a learning and interfaces algorithm that leverages weak supervision to construct the canonical space from task demonstrations. Next, we theoretically demonstrated why the model structure, training data, and loss functions used in PECAN help ensure that this canonical space is intuitive and user-friendly. In practice, our approach outputs a low-dimensional manifold; each point in the manifold corresponds to a style, and humans can specify their desired style across each task in the dataset by simply clicking on their preferred point. When experimentally compared to the alternatives, PECAN resulted in a more consistent interface that participants found easier to use over repeated interactions.

## REFERENCES

[1] Arthur Allshire, Roberto Martín-Martín, Charles Lin, Shawn Manuel, Silvio Savarese, and Animesh Garg. 2021. Laser: Learning a latent action space for efficient reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 6650–6656.

[2] Kareem Amin, Nan Jiang, and Satinder Singh. 2017. Repeated inverse reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1813–1822.

[3] Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. 2022. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research* 41, 1 (2022), 45–67.

[4] Erdem Bıyık, Malayandi Palan, Nicholas C Landolfi, Dylan P Losey, and Dorsa Sadigh. 2019. Asking easy questions: A user-friendly approach to active reward learning. In *Annual Conference on Robot Learning*. 1177–1190.

[5] Erdem Bıyık, Aditi Talati, and Dorsa Sadigh. 2022. Aprel: A library for active preference-based reward learning algorithms. In *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 613–617.

[6] Andreea Bobu, Andi Peng, Pulkit Agrawal, Julie A Shah, and Anca D Dragan. 2024. Aligning human and robot representations. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*. 42–54.

[7] Mahdi Bonyani, Maryam Soleymani, and Chao Wang. 2024. Style-Based Reinforcement Learning: Task Decoupling Personalization for Human-Robot Collaboration. In *International Conference on Human-Computer Interaction*. Springer, 197–212.

[8] Zhangjie Cao, Erdem Biyik, Woodrow Z. Wang, Allan Raventos, Adrien Gaidon, Guy Rosman, and Dorsa Sadigh. 2020. Reinforcement Learning based Control of Imitative Policies for Near-Accident Driving. In *Proceedings of Robotics: Science and Systems (RSS)*.

[9] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems* 30, 9 (2017), 4302–4310.

[10] Florian Graf, Christoph Hofer, Marc Niethammer, and Roland Kwitt. 2021. Dissecting supervised contrastive learning. In *International Conference on Machine Learning*. PMLR, 3821–3830.

[11] Yue Guo, Rohit Jena, Dana Hughes, Michael Lewis, and Katia Sycara. 2021. Transfer learning for human navigation and triage strategies prediction in a simulated urban search and rescue task. In *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 784–791.

[12] Donald Joseph Hejna III and Dorsa Sadigh. 2023. Few-shot preference learning for human-in-the-loop rl. In *Conference on Robot Learning*. PMLR, 2014–2025.

[13] Ashesh Jain, Shikhar Sharma, Thorsten Joachims, and Ashutosh Saxena. 2015. Learning preferences for manipulation tasks from online coactive feedback. *The International Journal of Robotics Research* 34, 10 (2015), 1296–1313.

[14] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*.

[15] Pallavi Koppol, Henny Admoni, and Reid G Simmons. 2021. Interaction Considerations in Learning from Humans.. In *IJCAI*. 283–291.

[16] Mengxi Li, Alper Canberk, Dylan P Losey, and Dorsa Sadigh. 2021. Learning human objectives from sequences of physical corrections. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2877–2883.

[17] Dylan P Losey, Andrea Bajcsy, Marcia K O'Malley, and Anca D Dragan. 2022. Physical interaction as communication: Learning robot objectives online from human corrections. *The International Journal of Robotics Research* 41, 1 (2022), 20–44.

[18] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. 2020. Learning latent plans from play. In *Conference on robot learning*. PMLR, 1113–1132.

[19] Zhao Mandi, Fangchen Liu, Kimin Lee, and Pieter Abbeel. 2022. Towards more generalizable one-shot visual imitation learning. In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2434–2444.

[20] Shaunak A Mehta and Dylan P Losey. 2023. Unified learning from demonstrations, corrections, and preferences during physical human-robot interaction. *ACM Transactions on Human-Robot Interaction* (2023).

[21] Thibaut Munzer, Marc Toussaint, and Manuel Lopes. 2017. Preference learning on the execution of collaborative human-robot tasks. In *IEEE International Conference on Robotics and Automation*. 879–885.

[22] Stefanos Nikolaidis, Ramya Ramakrishnan, Keren Gu, and Julie Shah. 2015. Efficient model learning from joint-action demonstrations for human-robot collaborative tasks. In *ACM/IEEE International Conference on Human-Robot Interaction*. 189–196.

[23] Takayuki Osa and Shuehi Ikemoto. 2020. Goal-conditioned variational autoencoder trajectory primitives with continuous and discrete latent codes. *SN Computer Science* 1, 5 (2020), 303.

[24] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. 2018. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 3758–3765.

[25] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. 2020. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems* 3 (2020), 297–330.

[26] Sascha Rosbach, Vinit James, Simon Großjohann, Silviu Homoceanu, and Stefan Roth. 2019. Driving with style: Inverse reinforcement learning in general-purpose planning for automated driving. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2658–2665.

[27] Dorsa Sadigh, Anca Dragan, Shankar Sastry, and Sanjit Seshia. 2017. Active preference-based learning of reward functions. In *Proceedings of Robotics: Science and Systems (RSS)*.

[28] Avi Singh, Eric Jang, Alexander Irpan, Daniel Kappler, Murtaza Dalal, Sergey Levinev, Mohi Khansari, and Chelsea Finn. 2020. Scalable multi-task imitation learning with autonomous improvement. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2167–2173.

[29] Marek Śmieja, Maciej Wołczyk, Jacek Tabor, and Bernhard C Geiger. 2020. Segma: Semi-supervised Gaussian mixture autoencoder. *IEEE transactions on neural networks and learning systems* 32, 9 (2020), 3930–3941.

[30] Jonathan Spencer, Sanjiban Choudhury, Matthew Barnes, Matthew Schmittle, Mung Chiang, Peter Ramadge, and Sidd Srinivasa. 2022. Expert intervention learning: An online framework for robot learning from explicit and implicit human feedback. *Autonomous Robots* (2022), 1–15.

[31] Matthew J Vowels, Necati Cihan Camgoz, and Richard Bowden. 2020. Gated variational autoencoders: Incorporating weak supervision to encourage disentanglement. In *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*. IEEE, 125–132.

[32] Nils Wilde, Alexandru Blidaru, Stephen L Smith, and Dana Kulić. 2020. Improving user specifications for robot behavior through active preference learning: Framework and evaluation. *The International Journal of Robotics Research* 39, 6 (2020), 651–667.

[33] Bryce Woodworth, Francesco Ferrari, Teofilo E Zosa, and Laurel D Riek. 2018. Preference learning in assistive robotics: Observational repeated inverse reinforcement learning. In *Machine learning for healthcare conference*. PMLR, 420–439.

[34] Bian Xihan, Oscar Mendez, and Simon Hadfield. 2022. SKILL-IL: Disentangling skill and knowledge in multitask imitation learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 7060–7065.

[35] Jerrold H Zar. 2005. *Spearman rank correlation*. Vol. 7. Wiley Online Library.

[36] Huixin Zhan, Feng Tao, and Yongcan Cao. 2021. Human-guided robot behavior learning: A gan-assisted preference-based reinforcement learning approach. *IEEE Robotics and Automation Letters* 6, 2 (2021), 3545–3552.

[37] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. 2020. robosuite: A Modular Simulation Framework and Benchmark for Robot Learning. *arXiv preprint arXiv:2009.12293* (2020).

[38] Mark Zolotas and Yiannis Demiris. 2022. Disentangled sequence clustering for human intention inference. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 9814–9820.