2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Diagnosing and Repairing Feature Representations under Distribution Shifts

Inês Lourenço, Andreea Bobu, Cristian R. Rojas, and Bo Wahlberg

***Abstract*— Robots have been increasingly better at doing tasks for humans by learning from their feedback, but still often suffer from model misalignment due to missing or incorrectly learned features. When the features the robot needs to learn to perform its task are missing or do not generalize well to new settings, the robot will not be able to learn the task the human wants and, even worse, may learn a completely different and undesired behavior. Prior work shows how the robot can detect when its representation is missing some feature and can, thus, ask the human to be taught about the new feature; however, these works do not differentiate between features that are completely missing and those that exist but do not generalize to new environments. In the latter case, the robot would detect misalignment and simply learn a new feature, leading to an arbitrarily growing feature representation that can, in turn, lead to spurious correlations and incorrect learning down the line. In this work, we propose separating the two sources of misalignment: we propose a framework for determining whether a feature the robot needs is incorrectly learned and does not generalize to new environment setups *vs.* is entirely missing from the robot's representation. Once we diagnose the source of error, we show how the human can initiate the realignment process for the model: if the feature is missing, we follow prior work for learning new features; however, if the feature exists but does not generalize, we use data augmentation to expand its training and, thus, complete the repair process. We demonstrate the proposed approach in experiments with a simulated 7DoF robot manipulator and physical human corrections.**

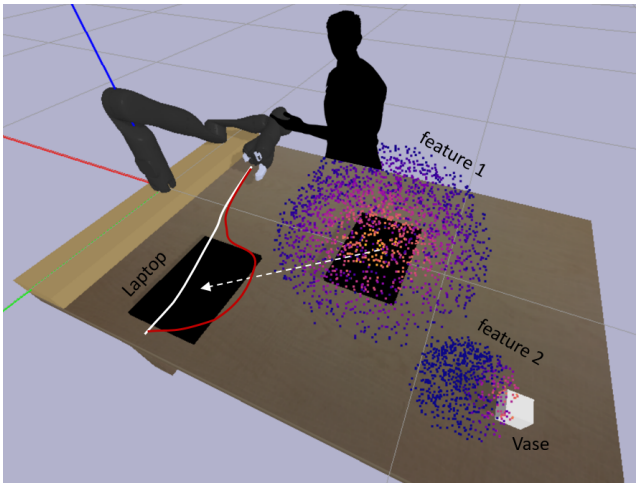

Fig. 1: In a new environment, a laptop is encountered in a position the robot has never seen it. It is, therefore, not sure which (if any) of the features it learned during training belong to it. By analyzing the corrections the human applies to its trajectory in the vicinity of the laptop, the robot *i)* infers the relation between its learned features and the object (diagnosis), and *ii)* adjusts the features to this new environment (repair).

I. INTRODUCTION

Communication is a key part of human life. When transmitting an idea to someone, we create *representations*, or *models*, of what we believe they are understanding or trying to make us understand. Communication flows naturally through this process until a behavior or response indicates that the model might be wrong. For instance, in Figure 1 the human wants the robot to transport a cup of coffee while staying away from the laptop, but the robot has never seen the laptop in this position – its model is different from the human's! At this point, an *intervention* is typically conducted in the form of asking for an explanation or justification for the unexpected behavior. This intervention, despite not being constantly necessary, becomes crucial for the mutual understanding of the circumstances and the *realignment* of the models. In our Human-Robot Interaction (HRI) example, the human can explain to the robot how to repair misaligned features by performing an intervention in the form of a physical correction.

The need for seamless and effortless interactions like this between humans and robots becomes fundamental as robots become increasingly more apt to perform tasks for and with humans. Misalignment happens when the robot's model of the human is incomplete, incorrect, or outdated, or when the human's model of the robot is similarly limited. It can lead to misunderstandings and ineffective interactions that are particularly undesirable when robots are expected to interact with humans in a natural and intuitive manner. The interventions can take the form of, albeit not being limited to, demonstrations [1], corrections [2], comparisons [3] and teleoperation [4].

Inverse reinforcement learning (IRL) is currently the most popular method for robots to estimate preferences and goals of those that surround them from their behavior, typically in the form of a reward or cost function. Traditional methods in IRL, however, assume that the human preferences are present

This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP), the Swedish Research Council Research Environment NewLEADS under contract 2016-06079, the Digital Futures project EXTREMUM, and Apple AI/ML Fellowship.

I. Lourenco, C. R. Rojas, and B. Wahlberg are with the Division of Decision and Control Systems, KTH Royal Institute of Technology, Stockholm, Sweden {ineslo,crro,bo}@kth.se

A. Bobu is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, CA, USA abobu@berkeley.edu

in the robot's representation. If this assumption does not hold, the cost function the robot learns from the human's demonstrations and physical corrections is defective.

As in the previous example, in this work we consider that humans can communicate their preferences to robots by intervening on their trajectories by means of physical corrections, entering the field of physical Human-Robot Interaction (pHRI). To convey to the robot that it should transport the cup of coffee far away from the laptop, the human can push the robot's arm to the side to prevent passing above it. If the robot has a feature related to the distance to the laptop in its feature representation, it will understand the correction, update its model, and replan the trajectory to continue further away from the laptop. But if it does not – or if the feature was not trained to generalize to this environment configuration – it will assign a wrong feature to try to explain the correction, hindering the learning process.

We build upon previous work based on detecting misalignment by estimating the robot's confidence in human corrections, only updating its knowledge if it is confident that it understands the correction. One advantage of this approach is avoiding learning from noisy actions when the human is not optimal. However, we argue that even when the human is optimal, the existence of misalignment associated with the robot's model can prevent the robot from understanding the correction. In that case, we propose that there is more information in the correction that the robot should take advantage of, in order to repair the misalignment.

One well-known example of representation misalignment is the shift in features at train vs. test time, due to the difficulty of models to generalize to different settings. This is referred to as distribution shift [5] and it is common in IRL since interventions require human effort which renders the generalization of the model to all unexplored settings challenging. Consider, for example, that there was a vase on top of the laptop during training. Now that the laptop is in a different position, should the learned features change with it, or are they instead related to the vase, which stayed in the same place? In other words, which objects did the features depend on? In this work, we enable robots to adapt to distribution shifts by learning from the relation between features and environment to disambiguate between incorrect features that do not generalize to new environments and completely missing features.

The main contributions of this work are:

- *Misalignment diagnosis* – when a robot detects misalignment in its feature representation by not being able to understand human input, we design a framework that enables it to identify which features are misaligned, and disambiguate between incorrectly learned features that do not generalize to new environment setups and entirely missing features from the robot's model.
- *Misalignment repair* – we propose a framework that enables robots to realign the misaligned features by using data augmentation to generalize them to the new task, and the missing features by asking the human for new data about the feature.
- We evaluate the two methods on a 7DoF simulated robotic arm that aims to align its representation to the human's, based on the physical torques received while performing a task.

The remainder of this paper is organized as follows: Section II gives an overview of related literature in the field, Section III formulates an IRL framework to detect confidence in the human input, and Section IV presents and discusses our proposed methods to diagnose and repair misaligned features. Finally, Section V evaluates how the proposed framework works in a 7-DoF robotic manipulator, and Section VI concludes with a discussion of some of the advantages and open problems of our framework, as well as suggestions for future research directions.

II. RELATED WORK

Inverse reinforcement learning is a popular framework for learning cost functions from human demonstrations [6], [7], which considers the human as a utility-driven agent that chooses actions with respect to an internal cost function. As an approach to Inverse Optimal Control that does not require a model of the environment, it is particularly useful when the cost functions are difficult or impractical to manually design.

Learning from corrections is another way of learning from human input that can be a good complement and advantageous in many situations where real-time and task-specific learning is needed [8], [9]. Methods to incorporate corrections in real-time to align robot and human preferences have been shown to improve performance and adaptability for HRI [2], [10], [11], [12].

Uncertainty in robot learning can be incorporated into the representations learned by maintaining a probability distribution over what the cost functions might be [13], [14]. In [15], the authors proposed using a Kalman filter to reason over the uncertainty of the estimated human preferences from physical corrections. However, even by keeping track of uncertainty, these works still assume that the human preferences lie in the robot's representation. In [16] and [17], a formulation is proposed where the robot's representation does not necessarily have to fully capture the human's underlying preferences. The authors propose learning the features proportionally to the robot's confidence in the human input, assuming low confidence to be a result of noisy or suboptimal human actions and leaving for future work expanding the robot's feature representation. In [18], the authors assume misalignment to be the result of missing features in the robot representation and solve it by querying the human for new input.

Generalization to new environments is a challenge in IRL which typically requires training from demonstrations rich enough to incorporate a wide variety of states of the environment. Methods like transfer learning [19] exist to adapt the learning in real-time or when the cost of retraining the model is prohibitively high. Our proposed framework fine-tunes the cost function to unexplored environments, by adjusting its features according to the changes in the environment.

III. PROBLEM FORMULATION

In this section we formalize the IRL framework used to teach a robot a cost function from offline feature demonstrations, how this cost function can be continuously improved from online human corrections, and how misalignment in the cost functions of the human and robot can be detected online by computing the robot's confidence in the human input.

A. Offline cost learning from demonstrations

IRL is a technique used to infer a cost function C from a given set of demonstrations $\mathcal{D} = \{\xi_1, \dots, \xi_D\}$. To make the problem tractable, C is typically parametrized by a vector $\theta \in \Theta$ representing the preferences of the human for how to perform the task, and the aim of the robot is to estimate these parameters. If θ was known by the robot, the problem could be treated as a Markov Decision Process (MDP). However, uncertainty over θ turns it into a Partially-Observable MDP (POMDP) formulation, with θ as a hidden part of the state. In this setting, the human actions are thus observations about θ under some observation model $P(\xi|\theta)$.

A popular decision-making model for human behavior is the Boltzmann model, which considers humans as noisily-optimal agents that typically choose control inputs that approximately minimize their cost [20], [21]. According to this model, the probability of giving a demonstration depends exponentially on its cost $C_\theta(\xi)$:

$$P(\xi|\theta) = \frac{e^{-C_\theta(\xi)}}{\int e^{-C_\theta(\bar{\xi})} d\bar{\xi}}. \quad (1)$$

By assuming $\{\xi_1, \dots, \xi_M\}$ *i.i.d.* and computing the maximum likehood with the Monte Carlo method, the estimated parameters $\hat{\theta}$ are then the ones that maximize the probability of the demonstrations:

$$\hat{\theta} = \arg\max_\theta \mathcal{L}(\theta) \approx D \log \sum_{i=1}^{D} e^{-C_\theta(\xi_i)} - \sum_{i=1}^{D} C_\theta(\xi_i). \quad (2)$$

Once an estimate of the cost function has been computed from training demonstrations, the robot then uses it to perform its task accordingly.

B. Online cost update from corrections

In order to enable the learning process of the robot to continuously adapt and adjust in real-time to different settings, work has been developed to apt it to learn from other sources of human input, such as physical corrections [2].

Formally, the problem can be formulated as a dynamical system $\dot{x} = f(x, u_R + u_H)$, where x is the state of the robot including its position and velocity, u_R is its action (e.g. the torque applied at the joints) and u_H the external torque applied by the human. There is a true objective function C_θ known by the human but not by the robot. One of the conventional models to approximate the infinite-dimensional space of possible cost functions is using basis functions [1], [7], based on which the cost can be written as a linear combination of features of the state, $C_\theta(x) = \theta^T \phi(x)$, which can be arbitrary mappings $\phi \colon \mathbb{R}^d \to [0, 1]$. While the relevant set of features ϕ is computed by the robot from feature demonstrations during training, the weights, θ, denoting the preferences of the human for performing the task, must be adjusted to the task at hand. We use this model henceforth throughout the paper. The correctional HRI framework is described next.

a) The robot acts: The robot uses its estimate of $\hat{\theta}$ obtained from (2) to compute a trajectory $\xi_R = \{x_t\}_{t=0}^T$ that minimizes its current cost function C_θ, and finds the control inputs $\{u_{R,t}\}_{t=0}^T$ to follow it. The cost of a trajectory is written as $C_\theta(\xi) = \theta^T \Phi(\xi)$, where $\Phi(\xi) = \sum_{x \in \xi} \phi(x)$ is the sum of the features ϕ along the trajectory ξ. The policy optimization scheme is given by

$$\xi_R = \arg\min_\xi \hat{\theta}^T \Phi(\xi). \quad (3)$$

b) The human corrects: If the robot follows a trajectory that does not seem correct to a noisily-rational human, they can, at a given time, choose to induce a joint torque u_H to deform the original trajectory to ξ_H. The deformed trajectory is given by $\xi_H = \xi_R + \mu A^{-1} U_H$ [2], where μ determines the magnitude of the deformation, the matrix $A \in \mathbb{R}^{T \times T}$ determines its shape, and $U_H = u_H$ at the moment of the correction and is 0 otherwise. The correction is done with the goal of minimizing the cost, while also minimizing human effort u_H. Hence, the observation model from (1) can be rewritten as

$$P(u_H|\xi_R; \theta) = \frac{e^{-(\theta^T \Phi(\xi_H) + \lambda ||u_H||^2)}}{\int e^{-(\theta^T \Phi(\bar{\xi}_H) + \lambda ||\bar{u}_H||^2)} d\bar{u}_H}, \quad (4)$$

where λ symbolizes the trade-off between cost and human effort.

c) The robot updates its knowledge: By computing the difference between the sum of the features of the original and deformed trajectories, the robot updates $\hat{\theta}$ as

$$\hat{\theta} \leftarrow \hat{\theta} - \alpha(\Phi(\xi_H) - \Phi(\xi_R)), \quad (5)$$

where $\alpha \geq 0$. More details on the derivations can be found in [2], but the intuitive interpretation is that the feature weights are updated based on the direction of the change of the feature values between the original and deformed trajectories. If the deformed trajectory passes further away from an object, the weights of the corresponding distance-to-object feature will increase.

C. Online misalignment detection

When a robot acts as described above, it takes the risk of naively trusting every human action, thus possibly learning from incorrect information. As mentioned in Section II, methods exist in the literature for keeping track of uncertainty, but a solution to the problem of whether the desired features exist in the robot's feature representation was, until recently, missing.

In [16], a method was presented to study when human corrections cannot be explained by the robot's model. The misalignment detection problem is tackled by adding a rationality coefficient $\beta \in [0, \infty)$ to (4), to account for uncertainty in how the human picks u_H. The computation of the two inference parameters, β and θ, can be separated into two parts and consists of analyzing how efficiently the feature

Algorithm 1 Misalignment detection

Require: ϕ_i for $i = 1, \ldots, M$
Require: $\xi_R = \arg\min_\xi \hat{\theta}^T \Phi(\xi)$, for initial $\hat{\theta}$.
1: **while** goal not reached **do**
2: **if** $u_H \neq \mathbf{0}$ **then**
3: $\xi_H = \xi_R + \mu A^{-1} \tilde{u}_H$
4: u_H^* computed from (6)
5: $\hat{\beta} = \frac{k}{2\lambda\left(\|u_H\|^2 - \|u_H^*\|^2\right)}$ } *Detect misalignment*
6: **if** $\hat{\beta}$ small **then**
7: *diagnose_and_repair_misalignment()* – Alg. 2
8: Recompute u_H^* and $\hat{\beta}$
9: **end if**
10: $\hat{\theta} \leftarrow \hat{\theta} - \alpha \frac{\Gamma(\Phi(\xi_H),1)}{\Gamma(\Phi(\xi_H),1)+\Gamma(\Phi(\xi_H),0)} \left(\Phi(\xi_H) - \Phi\left(\xi_R\right)\right).$
11: $\xi_R = \arg\min_\xi \hat{\theta}^T \Phi(\xi)$
12: **end if**
13: **end while**

values $\Phi(\xi_H)$ of the deformed trajectory can be explained by each of the robot's features. In a nutshell, the robot computes $\Phi(\xi_H)$ and searches for corrections, u_H^*, from the original trajectory, that could have achieved the same feature value change. The smallest correction is the one that the human would have performed if it was performing the correction due to dissatisfaction with that feature. This is formulated as the following constrained optimization problem:

$$\begin{aligned} \min_{\bar{u}_H} \quad & ||\bar{u}_H||^2 \\ s.t. \quad & \Phi(\xi_R + \mu A^{-1} \bar{u}_H) - \Phi(\xi_H) = 0. \end{aligned} \tag{6}$$

Then, by computing how far the optimal correction u_H^* is from the actual correction u_H received, the robot estimates its confidence $\hat{\beta}$ in the human input u_H:

$$\hat{\beta} = \frac{k}{2\lambda(||u_H||^2 - ||u_H^*||^2)}, \tag{7}$$

where k is the dimension of the action space.

Once we have an estimate of the confidence $\hat{\beta}$, we can compute the new posterior estimate of the human's cost function $p(\theta|\Phi(\xi_H), \hat{\beta})$, and from it obtain the update rule

$$\hat{\theta} \leftarrow \hat{\theta} - \alpha \frac{\Gamma\left(\Phi(\xi_H), 1\right)}{\Gamma\left(\Phi(\xi_H), 1\right) + \Gamma\left(\Phi(\xi_H), 0\right)} \left(\Phi(\xi_H) - \Phi\left(\xi_R\right)\right); \tag{8}$$

where

$$\Gamma\left(\Phi(\xi_H), i\right) = P(E = i \mid \hat{\beta}) P\left(\Phi(\xi_H) \mid \theta, E = i\right), \tag{9}$$

and E is a proxy variable for $\hat{\beta}$. The interested reader is referred to [16] for more details. It is, however, relevant to point out that if the possibility of the representations being misaligned is not taken into consideration, (8) simplifies to (5).

Once $\hat{\theta}$ is updated, the robot goes back to the task at hand.

In summary, the robot updates the parameters $\hat{\theta}$ of the cost function proportionally to its confidence $\hat{\beta}$ on the human correction. If some of its features can explain the correction, such as the human pushing the robot arm in the opposite direction of the laptop, $\hat{\beta}$ is large and the weight $\hat{\theta}$ of the distance-to-the-laptop feature is increased to represent

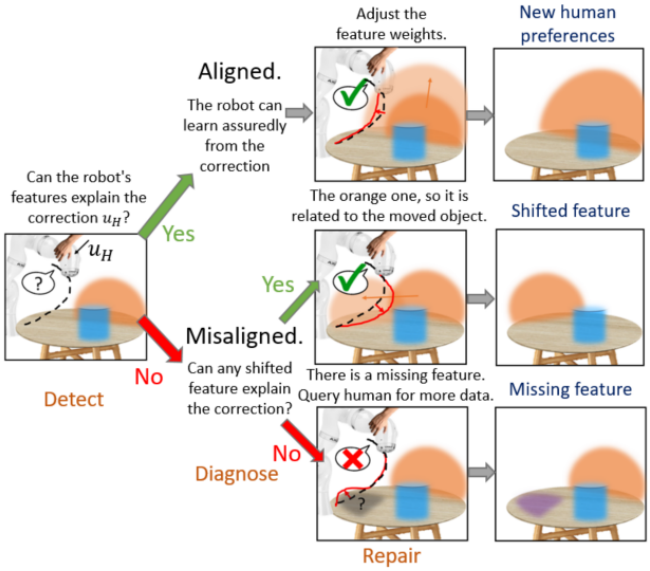

Fig. 2: Illustrative representation of detecting feature misalignment and diagnosing its cause using our framework, and the corresponding proposed solution to repair feature representations in each case.

that the human cares more about the distance to the laptop than previously estimated. If, on the other hand, $\hat{\beta}$ is small, such as in the case that the robot does not know about the distance-to-laptop feature, no values of $\hat{\theta}$ can make the cost function explain the correction; so $\hat{\theta}$ undergoes a proportionally small update. This confidence-based model update method is presented in Algorithm 1, and our proposed method in line 7 which delves into what the robot can learn from the corrections when the models are misaligned is presented in the next section.

IV. MODEL ALIGNMENT THROUGH FEATURE GENERALIZATION

In this section, we present how the cost function of the robot can be improved based on the human corrections, even when $\hat{\beta}$ is low. Recall that a low value of $\hat{\beta}$ means that there is no θ that can make the cost function $C_\theta(x) = \sum_{i=1}^{M} \theta_i \sum_{t=0}^{T} \phi_i\left(x^t\right)$ optimal in the light of the human corrections. Hence, the misalignment of the cost function has to be a result of a misaligned feature representation, $\phi_R = \{\phi_1, \ldots, \phi_M\}$; that is, $\phi_R \neq \tilde{\phi}_H$. It must be either the case that a feature is missing, $\exists\, \tilde{\phi}_i \in \tilde{\phi}_H$ s.t. $\phi_i \notin \phi_R$, or that at least one feature is incorrect, $\exists\, \tilde{\phi}_i \in \tilde{\phi}_H$ s.t. $\phi_i \in \phi_R$, but $\tilde{\phi}_i \neq \phi_i$.

The only existing relevant approach in the current literature to overcome representation misalignment is to query the human for more data, thus assuming that misalignment stems from the first case, which is an incomplete feature representation [18] ($\tilde{\phi}_i \notin \phi_R$). However, if the misaligned features actually exist in the robot's representation but were incorrectly learned and do not generalize to new environments ($\tilde{\phi}_i \neq \phi_i$), this assumption can lead to spurious correlations and incorrect learning down the line. To overcome this problem, in this section we propose a framework to

diagnose and repair model misalignment by differentiating between the two sources of misalignment.

First, one common reason for $\tilde{\phi}_H \neq \phi_R$ is the distribution shift problem, i.e., that ϕ_R consists of features learned for a specific training environment, which are not generalizable for new environments that the robot acts in, and, thus, become misaligned with the human's features $\tilde{\phi}_H$ in the new environment. The challenge of generalizing features to new settings stems from the fact that the features robots learn are a function of all the objects, and their respective positions, present in the environment at the moment of training, which makes the generalization problem highly dependent on large training datasets with different object positions.

Inspired by recent works on learning in real-time, in this section we propose an approach to instead address the generalization problem online as the robot performs the task, simply from human corrections. We tackle questions like: if the robot learns from feature demonstrations given when a vase is on top of a laptop, when one of the objects is shifted, should the feature learned from the demonstrations be shifted as well? When acting in different settings where the objects the robot has to interact with are in new positions, the problem becomes understanding i) how the learned features are related to the various objects, and ii) how they should be repaired when objects are shifted. We divide this two-fold problem into two main goals, which are each tackled in each of the next subsections, and conclude the section with some final remarks. The complete framework is schematically illustrated in Figure 2.

Problem 1 (Misalignment diagnosis). Once misalignment has been detected from (7), how can we identify if it stems from existing features that have not generalized to the new environment or completely missing features?

Problem 2 (Misalignment repair). After diagnosing the misaligned features, how can we align the representations by translating them to the new environment?

A. *Diagnosing misaligned features*

To make explicit the dependency of the state x on the surrounding environment, we define it as in [16] according to the position of the robot joints, and of the objects in the environment. In this way, $x = \{R, o^1, \dots, o^N\}$. Since the robot only has access to features $\phi = \{\phi_1, \dots, \phi_M\}$ trained in these states, we write them as $\phi(R, o^1, \dots, o^N)$. Due to the inexistence of a map between features and objects during training, the question then becomes how the robot can repair the features to $\tilde{\phi}(\tilde{R}, \tilde{o}^1, \dots, \tilde{o}^N)$, now adapted to the testing setting. Here, $\tilde{o}^i$ represents the new position of object i, and $\tilde{R}$ the new configuration of the robot.

When tasked with computing an optimal trajectory in the new environment, the robot starts by computing a value for how much each object has shifted, $\Delta_i = o^i - \tilde{o}^i$, for $i = 1, \dots, N$. Initially, the robot does not know which features are of which objects so it plans its trajectory according to (3), where Φ is the sum of the values of the trained features $\phi(R, o^1, \dots, o^N)$. If no corrections are received while it performs the task, the trained features did not belong to the shifted objects so the robot can correctly perform the tasks. If corrections are received but $\hat{\beta}$, computed according to (7), is large, then the features are still correctly represented and the robot just has to update their importance $\hat{\theta}$ for that task as in (8). If, on the other hand, $\hat{\beta}$ is small, no $\hat{\theta}$ can explain the corrections and therefore the feature representation $\phi(R, o^1, \dots, o^N)$ has to be repaired to $\tilde{\phi}(\tilde{R}, \tilde{o}^1, \dots, \tilde{o}^N)$.

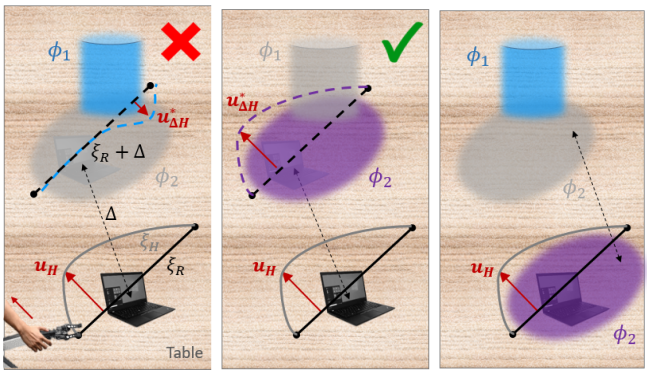

Fig. 3: During training, two features ϕ_1 and ϕ_2 are learned. At test time, the laptop is in a new position and the robot receives a correction u_H in its trajectory ξ_R. To diagnose the misalignment, we analyse what the optimal correction $u^*_{H\Delta}$ would have been for the previous laptop position in case the human was correcting for each of the features. (Left) The optimal correction $u^*_{H\Delta}$ for ϕ_1 would be very different from u_H, so $\hat{\beta}_\Delta$ is small. (Middle) The optimal correction for ϕ_2 would, on the other hand, be similar to u_H, so $\hat{\beta}_\Delta$ is large. ϕ_2 is thus a feature of the shifted object, and to repair the misalignment it must be shifted to its new position (Right).

To compute the misaligned features from the correction, we use a similar idea to that in Section III-C: we compute a confidence in how well each feature can explain the correction, but now taking into consideration that some features might have to be shifted if they belong to the shifted objects. We denote this new confidence β_{Δ_i}, and compute it according to

$$\hat{\beta}_{\Delta_i} = \frac{k}{2\lambda(||u_H||^2 - ||u^*_{H\Delta_i}||^2)}. \tag{10}$$

The parameters are as in Section III-C, but the comparison is now done between the human correction gotten, and the one that would have been optimal for that feature if it belonged to the shifted object.

The shifted optimal correction, $u^*_{H\Delta_i}$, is now computed by shifting the original and deformed trajectories by Δ_i as if object i was still in the same position as in training, and in that state computing the corresponding values for the trained features. In order words, we need to compute the feature values in a new state that was unexplored during training, but whose values might be the same, if the feature belongs to the shifted object, as those in a known training state. This can be formulated as the constrained optimization problem

$$\begin{aligned} \min_{\bar{u}_H} \quad & ||\bar{u}_H||^2 \\ s.t. \quad & \Phi(\xi_R + \Delta_i + \mu A^{-1}\bar{u}_H) - \Phi(\xi_H + \Delta_i) = 0. \end{aligned} \tag{11}$$

Algorithm 2 Misalignment diagnosis and repair

Require: $\xi_R, u_H, \xi_H, \phi(R, o^1, \ldots, o^N), \tilde{R}, \tilde{o}^1, \ldots, \tilde{o}^N$

$missing\_feature = 1$
for $\Delta_i = o^i - \tilde{o}^i$ **do** ▷ for each shift of an object
  for $\phi_1, \ldots, \phi_M$ **do** ▷ for each feature
    Compute $\Phi(\xi_R + \Delta_i)$, $\Phi(\xi_H + \Delta_i)$
    Solve (11) to get $u^*_{H\Delta_i}$
    Solve (10) to get $\hat{\beta}_{\Delta_i}$
    } *Diagnose misalignment*
    if $\hat{\beta}_{\Delta_i}$ large **then** ▷ ϕ_k is conditioned on o^i
      $inverse\_kinematics(EE + \Delta_i)$
      Repair ϕ to $\tilde{\phi}$ according to (12)
      missing_feature = 0
      } *Repair misalignment*
    end if
  end for
end for
if $missing\_feature == 1$ **then** ▷ entirely missing
  Query human for feature demos $\tilde{\phi}_{M+1}$ } *Repair misalignment*
end if

Figure 3 illustrates how (11) computes what the optimal correction for the different features would have been, if the object had not changed position. Hence, $\hat{\beta}_{\Delta_i}$ represents the confidence of the robot on the ability of the features to explain the correction, if these were shifted with the object.

An important condition must, however, be fulfilled.

Assumption 1. The corrections are performed by taking into consideration one feature at a time.

This assumption enables us to compare the contribution of each feature individually for the correction received. This is a common assumption in the literature, including in the methods in Section III, and has been shown to be reasonable – for example, [10] studied the advantages of correcting a robot performing IRL considering one feature at a time.

B. Repairing misaligned features

In the previous section we computed the confidence $\hat{\beta}_{\Delta_i}$ of the robot on each feature ϕ for each shift Δ_i. If this confidence is small for all the features, none of them can explain the correction received in the new environment. In this case, the misalignment has to derive from a missing feature, for which the robot was not trained. To repair this misalignment the robot proceeds by asking the human for data to learn the new feature. Details on how the cost function can be augmented by learning new features from neural networks can be found in [18].

If, on the other hand, the confidence $\hat{\beta}_{\Delta_i}$ is large for a certain feature under a certain shift, to repair the misalignment the robot must realign the feature to the new position of the object. This can be done according to

$$\tilde{\phi}_i(\tilde{R}, \tilde{o}^1, \ldots, \tilde{o}^N) \leftarrow \phi_i(R, o^1 - \Delta_1, \ldots, o^N - \Delta_N), \quad (12)$$

where $\tilde{\phi}_i$ is the updated feature ϕ_i, $\tilde{o}^i$ is the new position of the shifted object, and $\tilde{R}$ is the new position of the robot joints, since the features are a function of not only the object locations but also of the joint positions of the robot. The new join positions $\tilde{R}$ can be computed using standard inverse kinematics methods for a new end-effector (EE) position shifted by Δ_i.

The complete diagnosis and alignment framework is summarized in Algorithm 2. Once the features are repaired, the robot completes Algorithm 2 and returns to Algorithm 1. The weights of the newly aligned features still need, however, to be adjusted to the new environment. So from the same correction u_H, as before, the robot estimates the new u^*_H and $\hat{\beta}$. Based on these it then updates the weights $\hat{\theta}$ according to (8), and can resume its trajectory according to (3), now for Φ being the sum of the values of the aligned features $\tilde{\phi}(\tilde{R}, \tilde{o}^1, \ldots, \tilde{o}^N)$ from (12).

C. Remarks

Let us now discuss some details about the framework.

Multi-object dependent features: The proposed alignment framework presented in the previous subsections can be directly applied to features that depend on multiple objects, including concepts such as *distance between objects*, *above*, *near*, *aligned*, etc. (more examples can be found in [22]). If one of the objects is shifted in the test setting, like in the case of the vase that was *above* the laptop during training but moved, the *above* feature will not be relevant in the new setting so the human will not correct the robot trajectory near the moved vase object. The representation is therefore not altered. Further, in the position where the objects were during training, the robot will now be corrected to assign a small weight θ to the feature.

Object uncertainty: We assumed that the robot knows which objects have changed position. If this is not the case, Δ_i is not uniquely known but the procedure can still be repeated for multiple Δ_i's corresponding to the distance to each of the training objects until the correct one is found.

New inference objects: Despite having been presented for the case where objects are shifted between training and testing settings, the framework can also be directly applied to settings where new objects are present at testing time. Both Δ_i and β_{Δ_i} are computed as before, and based on them the robot evaluates if any of the features of the training objects apply to the new object. The difference is that instead of altering feature ϕ_i as in (12), a new one ϕ_{M+1}, is added. If the new objects are the same or behavior-invariant versions of the training objects (that is, the distinguishing characteristics, such as color, do not change how the object should be interacted with and therefore their features) – for example, if a mobile phone is seen at testing but not training, the robot could associate it with technology and therefore act as it would with the computer. This assumption could then be confirmed or denied by the human corrections received. This can be extended by computing a probability $P(\tilde{o}^{N+1} = o^i)$ of the new observations being behavior-invariant versions of the known ones), then

$$\tilde{\phi}_{M+1}(\tilde{R}, \tilde{o}^1, \ldots, \tilde{o}^N) = \phi_i(R, o^1 - \Delta_1, \ldots, o^N - \Delta_N). \quad (13)$$

If they are completely new objects not seen during training, $\tilde{\phi}_{M+1}$ is constructed from querying the human for new feature demonstrations.

Small and large $\hat{\beta}$: The magnitude of the confidence parameters $\hat{\beta}$ and $\hat{\beta}_{\Delta_i}$ is crucial for defining misalignment. We tune a threshold value offline by making use of the binary variable $E = \{0, 1\}$ defined in (9) as a proxy to $\hat{\beta}$. In [17], this threshold is computed via the Bayesian framework $P(E \mid \hat{\beta}) \propto P(\hat{\beta} \mid E)P(E)$ by fitting the distributions $P(\hat{\beta} \mid E)$ to data collected from controlled user interactions.

Complexity and real-time capability: Our algorithm scales bilinearly with the number of features and of objects shifted, and is applicable in any scenario with full observability where learning representations is a central problem and which can allow for real-time corrections during task execution. This can be extended from physical corrections to other types of corrections such as comparisons and these can potentially be given after task completion. Practical examples can go from shared autonomy for assistive robotics to computer vision for autonomous driving.

V. EXPERIMENTS

We evaluate our framework in a 7-DoF simulated JACO robotic manipulator implemented in Pybullet in a 1.80 GHz CPU. We added the ability for humans to correct the robotic arm trajectory by applying a torque with the cursor. For practical implementation details of learning from corrections in this setting, the reader is referred to [17, Appendix A]. We study the behavior of the robot in an environment with two objects: o^1, a black rectangle representing a *laptop*, and o^2, a white cube representing a *vase*; and two features: ϕ_1 : *distance-to-laptop*, a large point cloud in the center; and ϕ_2 : *distance-to-vase*, a smaller point cloud on the bottom left corner. The feature distances are computed from the position of the end-effector of the robot to the center of the object. Recall, however, that each feature $\phi_i(R, o^1, o^2), i = 1, 2$ is learned as a function of the robot position and both objects in the environment, meaning that the robot does not know which features belong to which objects.

At test time, the robot needs to transport a cup of coffee across the table, in a new environment where, for example, the laptop is in a different position $\tilde{o}^1$. Using TrajOpt [23], the optimal trajectory ξ_R is computed and followed according to (3), based on the trained features for the previous object positions. Let us consider the case where the cup should be transported in the vicinity of the position of the vase. In this new environment, the human prefers the robot to keep a bigger distance from it than during training, so it slightly pushes the arm away from the vase to change the trajectory of the robot. The robot follows Algorithm 1 and, through $\hat{\beta}$ from (7), computes that its current features (in particular, the distance-to-vase) can explain the correction, so increases the weight θ of this feature translating that it learned to stay further away from the vase. In the rest of the section, we analyze how our proposed framework performs when, on the other hand, $\hat{\beta}$ is small so the correction cannot be explained by any of the existing features.

Let us now analyze the case where the trajectory passes in the vicinity of the new laptop position, illustrated in Figure 4. Since, unlike the robot, the human knows that the

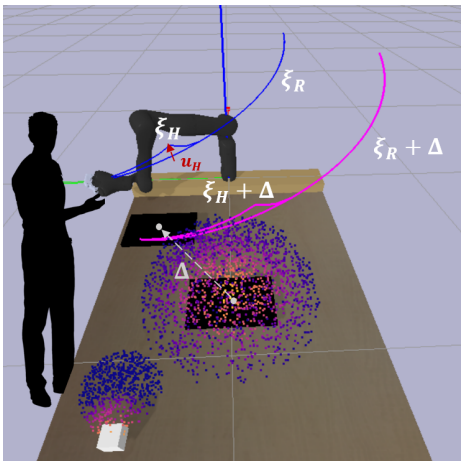

Fig. 4: If the human correction u_H cannot be explained by the current features, the robot translates the original and deformed trajectories ξ_R and ξ_H to the previous position of the object and computes how well each of the features in that state could explain the correction.

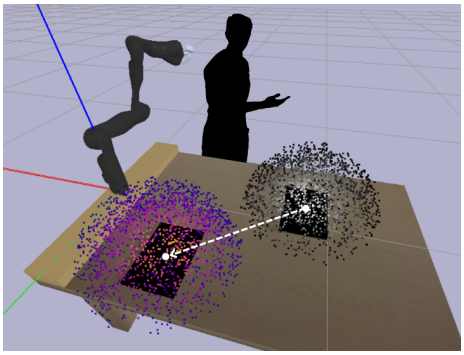

Fig. 5: Once a misaligned feature is identified, the robot can successfully repair it according to the new object position.

distance-to-laptop feature should be taken into account when performing a trajectory near the laptop, independently of its position, it applies a torque u_H to correct the trajectory to ξ_H to pass further away from it. Since none of the robot's current features were trained for those states, they cannot explain the correction so the robot follows Algorithm 2 to estimate $\hat{\beta}_\Delta$ as in (10). It computes what the values Φ for the two trained features would be for the training position of the laptop $\xi_R + \Delta$, as in Figure 3. In this case, $\Phi(\xi_R) = \Phi(\xi_H) = [0, 0]^T$, $\Phi(\xi_R + \Delta) = [20.13, 0]^T$ and $\Phi(\xi_H + \Delta) = [18.6, 0]^T$. From (11), it then computes the optimal correction u_H^* that would have been applied to that shifted trajectory according to each of the features. The second elements are always zero because the distance-to-vase feature is far away from all four trajectories. Therefore, $u_H^* = 0$ since the cost of that feature is already minimized, and the human has a preference for smaller corrections that minimize their effort (4). For the distance-to-laptop feature, on the other hand, the cost of the shifted deformed trajectory is decreased, resulting in a $u_H^* > 0$, and equal to u_H. Therefore, from (10) we get a small $\hat{\beta}_\Delta$ for the former

feature, but a large one for the latter, showing that the distance-to-laptop feature explains the correction performed near the new laptop position.

Once the robot learns that the distance-to-laptop feature corresponds to the object shifted, we apply the framework from Section IV-B to repair the feature values for the new setting, $\tilde{\phi}(\tilde{R}, \tilde{o}^1, o^2)$. Figure 5 shows how the feature values change in the vicinity of the new (in purple) and previous (in grey) positions of the object.

We evaluated $\hat{\beta}$ and $\hat{\beta}_\Delta$ over multiple experiments with random object shifts and robot trajectories, and the major limiting factor was the new position of the object being close to its original training position. Overall, it took around 0.5 seconds per feature and per object shifted to diagnose the misalignment (compute $\hat{\beta}_\Delta$) and repair (shift) the misaligned features. Compared to teaching the different features for multiple object positions during training, the amount of human effort prevented with this method can be estimated from data like [18, Figure 4]. Due to lack of space, implementation details, a more detailed analysis, and more complex examples will be linked in an extended version of this paper.

VI. CONCLUSIONS

In this work we argued that robots' adaptation capabilities should go beyond detecting when their representations are misaligned with those of humans. We proposed a framework that enabled them to diagnose the causes of the misalignment – by distinguishing between misalignment caused by incorrect features that do not generalize to new environments, and completely missing ones – and a method to repair the misalignment in each of these scenarios. While the latter cause of misalignment can be addressed by querying the human for more feature data, solving the former required estimating which objects the different learned features are related to. Our framework has the advantage of being applicable while the robot perfoms a task in real-time, leveraging information from physical human corrections. In simulations we showed that a robotic arm, trained to perform tasks for specific states, could successfully use our framework to diagnose, and augment, its misaligned features for new states in a new environment.

A. Future work

In this paper we took a step toward fully generalizable HRI. Solving model misalignment is a complex problem that can result from varied misalignment sources. While we distinguish between two – missing and incorrect features that do not generalize to new environments – others, such as features that have actually been learned wrongly due to limited training data, remain a question to address in future work. In the future, we would also like to present an extended study of each of the remarks in Section IV-C, and consider the possibility of allowing the robot to query the human in case of ambiguity about which feature should be repaired.

REFERENCES

[1] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2004.

[2] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan, "Learning robot objectives from physical human interaction," in *Conference on Robot Learning (CoRL)*, 2017, pp. 217–226.

[3] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[4] S. Javdani, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization," *Robotics: Science and Systems (RSS)*, 2015.

[5] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, *Covariate shift and local learning by distribution matching*. MIT Press, Cambridge, MA, USA, 2009.

[6] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning." in *International Conference on Machine Learning (ICML)*, 2000, p. 663–670.

[7] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *Conference on Artificial Intelligence (AAAI)*, 2008, pp. 1433–1438.

[8] I. Lourenço, R. Mattila, C. R. Rojas, and B. Wahlberg, "Cooperative system identification via correctional learning," *19th IFAC Symposium on System Identification*, vol. 54, no. 7, pp. 19–24, 2021.

[9] I. Lourenço, R. Winqvist, C. R. Rojas, and B. Wahlberg, "A teacher-student markov decision process-based framework for online correctional learning," in *IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 3456–3461.

[10] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan, "Learning from physical human corrections, one feature at a time," in *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2018, pp. 141–149.

[11] A. Jain, S. Sharma, T. Joachims, and A. Saxena, "Learning preferences for manipulation tasks from online coactive feedback," *International Journal of Robotics Research (IJRR)*, vol. 34, no. 10, pp. 1296–1313, 2015.

[12] R. A. Gutierrez, V. Chu, A. L. Thomaz, and S. Niekum, "Incremental task modification via corrective demonstrations," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1126–1133.

[13] D. Hadfield-Menell, S. Milli, P. Abbeel, S. J. Russell, and A. Dragan, "Inverse reward design," *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[14] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning." in *International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 7, 2007, pp. 2586–2591.

[15] D. P. Losey and M. K. O'Malley, "Including uncertainty when learning from human corrections," in *Conference on Robot Learning (CoRL)*, 2018, pp. 123–132.

[16] A. Bobu, A. Bajcsy, J. F. Fisac, and A. D. Dragan, "Learning under misspecified objective spaces," in *Conference on Robot Learning (CoRL)*, 2018, pp. 796–805.

[17] A. Bobu, A. Bajcsy, J. F. Fisac, S. Deglurkar, and A. D. Dragan, "Quantifying hypothesis space misspecification in learning from human–robot demonstrations and physical corrections," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 835–854, 2020.

[18] A. Bobu, M. Wiggert, C. Tomlin, and A. D. Dragan, "Feature expansive reward learning: Rethinking human input," in *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2021, pp. 216–224.

[19] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[20] J. Von Neumann and O. Morgenstern, in *Theory of games and economic behavior*. Princeton University Press, 1944.

[21] C. L. Baker, J. B. Tenenbaum, and R. R. Saxe, "Goal inference as inverse planning," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 29, no. 29, 2007.

[22] A. Bobu, C. Paxton, W. Yang, B. Sundaralingam, Y.-W. Chao, M. Cakmak, and D. Fox, "Learning perceptual concepts by bootstrapping from human queries," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 260–11 267, 2022.

[23] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: Science and Systems (RSS)*, vol. 9, no. 1, 2013, pp. 1–10.