An Incremental Inverse Reinforcement Learning Approach for Motion Planning with Separated Path and Velocity Preferences

Avaei, S.; van der Spaa, L.F.; Peternel, L.; Kober, J.

DOI
10.3390/robotics12020061

Publication date
2023

Document Version
Final published version

Published in
Robotics

robotics

MDPI

Article

An Incremental Inverse Reinforcement Learning Approach for Motion Planning with Separated Path and Velocity Preferences

Armin Avaei [1,†], Linda van der Spaa [1,2,*,†], Luka Peternel [1] and Jens Kober [1]

[1] Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands; arminavaei@gmail.com (A.A.); l.peternel@tudelft.nl (L.P.); j.kober@tudelft.nl (J.K.)
[2] Honda Research Institute Europe, 63073 Offenbach am Main, Germany
* Correspondence: l.f.vanderspaa@tudelft.nl
† These authors contributed equally to this work.

Abstract: Humans often demonstrate diverse behaviors due to their personal preferences, for instance, related to their individual execution style or personal margin for safety. In this paper, we consider the problem of integrating both path and velocity preferences into trajectory planning for robotic manipulators. We first learn reward functions that represent the user path and velocity preferences from kinesthetic demonstration. We then optimize the trajectory in two steps, first the path and then the velocity, to produce trajectories that adhere to both task requirements and user preferences. We design a set of parameterized features that capture the fundamental preferences in a pick-and-place type of object transportation task, both in the shape and timing of the motion. We demonstrate that our method is capable of generalizing such preferences to new scenarios. We implement our algorithm on a Franka Emika 7-DoF robot arm and validate the functionality and flexibility of our approach in a user study. The results show that non-expert users are able to teach the robot their preferences with just a few iterations of feedback.

Keywords: learning from demonstration; human preferences; incremental inverse reinforcement learning; coactive learning; physical human–robot interaction

Citation: Avaei, A.; van der Spaa, L.; Peternel, L.; Kober, J. An Incremental Inverse Reinforcement Learning Approach for Motion Planning with Separated Path and Velocity Preferences. *Robotics* **2023**, *12*, 61. https://doi.org/10.3390/robotics12020061

Academic Editor: Dan Zhang

Received: 7 March 2023
Revised: 1 April 2023
Accepted: 14 April 2023
Published: 20 April 2023

1. Introduction

Autonomy is increasingly being discussed regarding cooperation. A gentler breed of robots, "cobots", have started to appear in factories, workshops and construction sites, working together with humans. A challenge in the deployment of such robots is producing desirable trajectories for object carrying tasks. A desirable trajectory not only meets the task constraints (e.g., collision-free movement from start to goal), but also adheres to user preferences. Such preferences may vary between users, environments and tasks. It is infeasible to manually encode them without exact knowledge of how, with whom and where the robot is being deployed [1]. Manual programming is even more detrimental in cooperative environments, where robots are required to be easily and rapidly reprogrammed. In this context, learning preferences directly from humans emerges as an attractive solution.

We address the challenge of learning personalized human preferences, starting from a robot plan that may not match the execution style or safety standards of a specific human user (e.g., robot carries the object closer to the obstacle than the user prefers). Figure 1 illustrates how a user may demonstrate a trajectory encoding multiple implicit preferences to correct the original robot plan.

One way to adhere to human preferences is by means of variable impedance control [2,3]. Although such strategies can ensure safe and responsive adaptation, they suffer from being purely reactive (i.e., they do not remember the corrections). The robot should not only conform to a new trajectory, but it has to update its internal model in order to understand the improvements in the corrected trajectory [4–6]. Thus, ideally, we should encode knowledge

of humans' desired trajectories as a set of parameters that are incrementally updated based on the corrected trajectory.

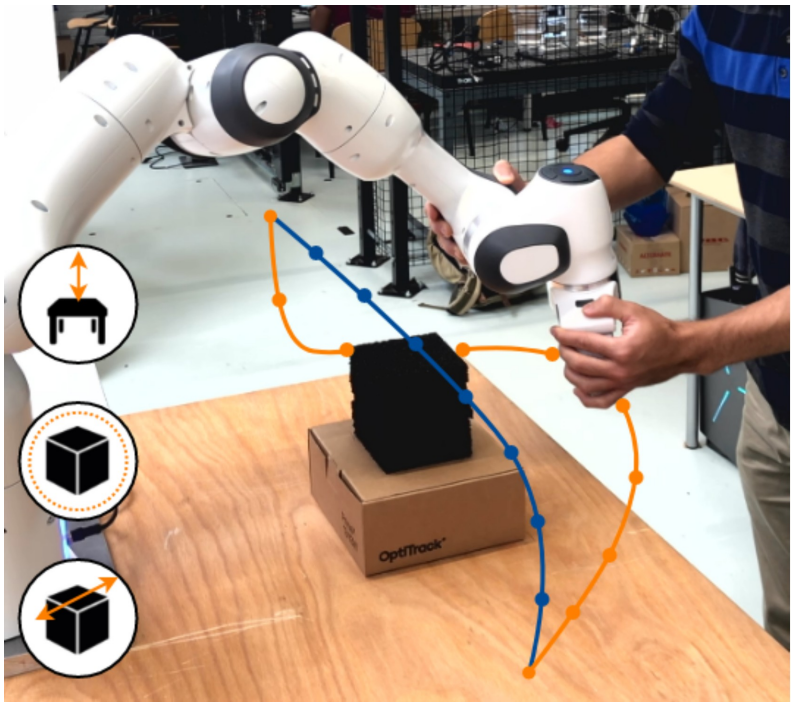

Figure 1. Leveraging demonstrations as means of understanding the human's preferences in an object carrying task: The robot originally plans the blue trajectory without knowledge of human preferences. The user demonstrates the orange trajectory which in this instance contains the following preferences: "stay close to the table surface", "maintain a larger distance from the obstacle" and "pass on the far side of the obstacle". We develop a method for learning and generalizing such preferences to new scenarios (i.e., new start, goal or obstacle positions).

To this end, the Learning from Demonstration (LfD) approach enables robots to encode human-demonstrated trajectories. LfD frameworks have the advantage of enabling non-experts to naturally teach trajectories to robots. A widespread trajectory learning method in LfD is Dynamic Movement Primitives (DMPs) [7]. In addition to encoding trajectories, DMPs are able to adapt the learned path by updating an interactive term in the model [8,9]. Additionally, they can adapt the velocity of the motion by estimating the frequency and the phase of a periodic task [3] or learning a speed scaling factor [10]. As a result, DMPs can capture human path and velocity preferences on a trajectory level. Losey and O'Malley [5] demonstrated that such velocity preferences can also be learned online from interactive feedback, although with some effort. However, these methods lack any knowledge about the task context or why the trajectory was adjusted in the first place. Hence, such an approach fails to generalize user preferences to new scenarios due to the lack of a higher-level understanding of human actions.

A better approach is to pair parameters with features that capture contextual information (e.g., distance to obstacle) and utilizes this information to find an optimal solution in new scenarios. Such generalizations can be achieved by learning a model of what makes a trajectory desirable. Modeling assumptions can be made to form a conditional probability distribution over trajectories and contextual information, e.g., as demonstrated by Ewerton et al. [11]. Although this has been proven effective in simple reaching tasks, whether such models can directly capture complex human preferences in a contextually rich environment remains an open question. However, Inverse Reinforcement Learning (IRL) approaches have already proven to be capable of this [12].

Unlike traditional IRL methods requiring expert demonstrations [13,14], more recently derived algorithms allow preference learning from user comparisons of sub-optimal

trajectories [12]. Potentially, a much wider range of human behavior can be interpreted as feedback for preference learning in general [15]. In this paper, however, we focus on reward learning for robot trajectories. A model-free approach can be used to learn complex non-linear reward functions [16], but such an approach requires many queries to learn from, which is time-intensive. Therefore, we keep a simple linear reward structure. To shape this reward, we identified four fundamental preference features of the pick-and-place type of object transportation tasks in the literature: height from table/ground [1,4,6], distance to obstacle [1,17], obstacle side [18,19] and velocity [3,10]. These features are relatively scenario-unspecific, and are therefore suitable for generalization in object transportation tasks of the kind we consider in this paper: pick-and-place tasks in the presence of obstacles. To the best of our knowledge, there is no method to account for all these features together in a unified framework.

Given such a set of features, coactive learning [20] can be used to learn a reward function. In coactive learning, the learner and the teacher both play an active role in the learning process; the learner proposes one or multiple solutions and learns from the relative feedback provided by the teacher in response. Coactive learning has an upper boundary on regret, leaving room for noisy and imperfect user feedback. Furthermore, it is an online algorithm, i.e., the system can learn incrementally from sequential feedback. An adapted version of coactive learning was applied by Jain et al. [1] to learn trajectory preferences in object carrying tasks. To this end, users iteratively ranked trajectories proposed by the system. Although selected based on the learned reward, the trajectories were generated using randomized sampling, which increases the number of feedback iterations necessary for convergence. Methods by Bajcsy et al. [4] and Losey et al. [6] adapt the robot trajectory to the user's preferences based on force feedback and optimize the remaining trajectory with online correction in a specific scenario. However, these methods cannot capture velocity preferences on top of path preferences.

To address this gap in the state-of-the-art methods, we propose a novel framework for optimizing trajectories in object transportation tasks that meet the user's path and velocity preferences, where we first optimize the path and then the velocity on the path. The objective function for the optimization comprises a human preference reward function and a robot objective function that ensures the safety and efficiency of the trajectories. This explicit separation of the agents' objectives allows for negotiation, where the robot is recognized as an intelligent agent which may give valuable input of its own.

The approach takes a full demonstrated trajectory as the feedback for the learning model, comparing it to the robot's previous plan at each iteration. A minimum acceleration trajectory model significantly reduces the size of the task space, hence increasing the optimization efficiency. To capture the preferences, we design a set of features that correspond to the four preferences, covering both the motion shape and timing, which we identified from the literature to be fundamental for the considered pick-and-place tasks.

Unlike Bajcsy et al. [4] and Losey et al. [6], we request iterative feedback and employ an optimization scheme that samples from the global trajectory space. Although this is less efficient in terms of human effort for teaching preferences in a specific scenario (i.e., the user has to provide at least one full task demonstration), it allows us to additionally capture velocity preferences on top of the path preferences. Furthermore, our method enables the separation of velocity and path preferences both during the learning and in the trajectory optimization stage. With our combination of a trajectory optimization scheme and carefully selected preference features, we can generalize to new contexts without needing (many) additional corrective demonstrations. In contrast to work by Jain et al. [1], we learn from a few informative feedback demonstrations and give special attention to the trajectory sampling by employing model-based trajectory optimization. This facilitates fast learning and generalization of preferences to entirely new contexts.

We evaluate the proposed method in a user study on a 7-DoF Franka Emika robot arm. In the key previous user studies of learning human preferences [4,6,21], the experimenter instructed the human participants what preference to demonstrate to the robot.

In contrast, in our user study, we let the participants freely select their own preferences while demonstrating the task execution to the robot. Additionally, our study examines whether the users can actually distinguish the learned trajectory capturing their preference from the trajectories capturing only part of their preference. In a supplementary study, we qualitatively compare our method to two relevant methods from the literature. We discuss the structural differences between the methods and show by simulation how these differences affect the learning of preferences from human (corrective) demonstrations.

In summary, this paper's main contribution is a methodology that is able to capture velocity preferences on top of path preferences by separating the velocity optimization from the path optimization. Learning the path and velocity separately provides users with the option to avoid the challenge of providing a temporally consistent demonstration at each iteration. This offers users the flexibility to demonstrate their path and velocity preferences either simultaneously or in separate demonstrations. Secondly, the learned preferences are transferred to new scenarios by exploiting a trajectory model. Importantly, we perform a user study to validate whether the proposed method can learn and generalize freely chosen preferences, in contrast to the many user studies in the literature which prescribe user preferences. Additionally, we perform a supplementary study to compare the pros and cons of the proposed approach with two common methods from the literature.

The rest of the paper is organized as follows: In Section 2, we explain the algorithm and methodology in detail. The user study is described in Section 3 and the experimental results are also shown and discussed. A supplementary study is presented and discussed in Section 4. Finally, we present our conclusions and a view on future work in Section 5.

2. Method

The problem is defined in the following manner: given a context $\mathcal{C}$ describing start, goal and obstacle positions, the robot has to determine the trajectory $\boldsymbol{\xi} = [\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_N] \in \Xi$ (set of state sequences) that conforms to the human preferences and meets the task goals. The states are defined as $\boldsymbol{s}_k = [\boldsymbol{x}_k; \dot{\boldsymbol{x}}_k]$ (position and velocity), with k indicating trajectory samples.

In our setting, the true reward functions are known by the user but not directly observable by the robot. Hence, the problem can be seen as a Partially Observable Markov Decision Process (POMDP) [4]. Our reward functions have parameters that are part of the hidden state, and the trajectories provided by the user are observations about these parameters. Solving such problems, where the control space is very complex and highly dimensional, is challenging. Therefore, we simplify the problem through approximation of the policy by separating planning and control and treating it as an optimization problem. Furthermore, we make the problem tractable by reducing our state space to one of viable smooth trajectories.

The resulting framework, depicted in Figure 2, first learns the appropriate reward functions, then plans a trajectory maximizing the rewards via optimization. Once the trajectory is defined, we use impedance control to track it in a safe manner. Notably, we separate the problem of path and velocity planning in the learning and optimization steps. Updating the path and velocity weights separately provides users with the option to avoid the challenge of providing a temporally consistent demonstration at each iteration. As a result, users have the flexibility to demonstrate their path and velocity preferences either simultaneously or in separate demonstrations.

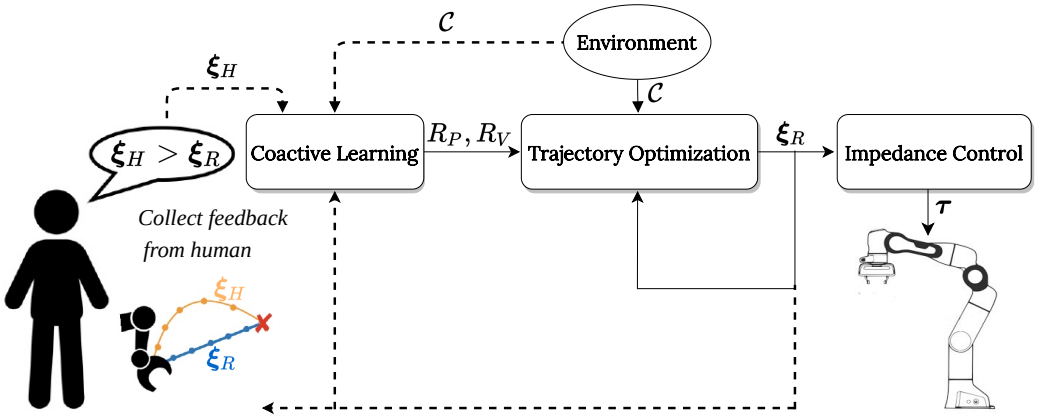

Figure 2. The human user provides demonstrations, which are used to learn a distribution over reward function via coactive learning. We use the learned rewards to optimize the robot's trajectory according to human preferences. The resulting trajectory is executed using an impedance controller. We repeat this process, querying the human for preferred trajectories until convergence. The human can then be taken out of the loop.

2.1. Learning Human Reward Functions from Demonstration

We follow previous IRL work [1,13] in assuming that the reward functions are a linear combination of features ϕ with weights θ. Accordingly, we define path and velocity reward functions R_P and R_V as

$$R_P(\boldsymbol{x}; \mathcal{C}, \boldsymbol{\theta}_{HP}) = \boldsymbol{\theta}_{HP}^T \boldsymbol{\Phi}_P(\boldsymbol{x}; \mathcal{C}), \tag{1}$$

$$R_V(\bar{x}, \bar{\boldsymbol{x}}; \mathcal{C}, \boldsymbol{\theta}_{HV}) = \boldsymbol{\theta}_{HV}^T \boldsymbol{\Phi}_V(\bar{x}, \bar{\boldsymbol{x}}; \mathcal{C}), \tag{2}$$

where $\boldsymbol{\theta}_{HP}$ and $\boldsymbol{\theta}_{HV}$ denote the unknown weights that, respectively, capture the human path and velocity preferences. In the case of the velocity reward, we divide the trajectory into equal segments (i.e., a range of samples) indicated by r. Then, $\bar{\boldsymbol{x}}_r$ and $\bar{x}_r$ are the average of the position vectors and the velocity norms in a segment. $\boldsymbol{\Phi}_P$ and $\boldsymbol{\Phi}_V$ are the total path and velocity feature counts along the trajectory:

$$\boldsymbol{\Phi}_P(\boldsymbol{x}; \mathcal{C}) = \sum_{k=1}^{N} \boldsymbol{\phi}_P(\boldsymbol{x}_k; \mathcal{C}), \; \boldsymbol{\Phi}_V(\bar{x}, \bar{\boldsymbol{x}}; \mathcal{C}) = \sum_{r=1}^{M} \boldsymbol{\phi}_V(\bar{x}_r, \bar{\boldsymbol{x}}_r; \mathcal{C}). \tag{3}$$

Note that the velocity features are a function of both the segment's velocity and position, allowing us to capture position-dependent velocity preferences.

To have comparable rewards, all trajectories are re-sampled to contain a fixed number of N states. The velocity inherently affects the number of samples within a trajectory, which is why we divide the trajectory into M segments and consider the average velocity within each segment ($M < N$). Features are directly computed from the robot state and context of the task. We describe them in the next subsection.

During kinesthetic demonstration, the robot is in gravity compensation mode. That gives the human full control over the demonstrated trajectories, which we assume to correlate exponentially to the human's internal reward:

$$P(\boldsymbol{\xi}_H | \mathcal{C}, \boldsymbol{\theta}_{HP}, \boldsymbol{\theta}_{HV}) \propto e^{\boldsymbol{\theta}_{HP}^T \boldsymbol{\Phi}_P(\xi_H; \mathcal{C}) + \boldsymbol{\theta}_{HV}^T \boldsymbol{\Phi}_V(\xi_H; \mathcal{C})}, \tag{4}$$

which, for brevity, we can write as $P(\boldsymbol{\xi}_H | \mathcal{C}, \boldsymbol{\theta}_H) \propto e^{\boldsymbol{\theta}_H^T \boldsymbol{\Phi}(\xi_H; \mathcal{C})}$.

Assuming that the human behavior is approximately optimal with respect to the true reward (i.e., their preferences), we use a variant of coactive learning introduced by Bajcsy et al. [4] to learn the weights $\boldsymbol{\theta}_{HP}, \boldsymbol{\theta}_{HV}$. However, we can only compute $\boldsymbol{\Phi}_P, \boldsymbol{\Phi}_V$ (3) over full trajectories. Therefore, instead of updating the weights based on an estimate of the human's intended trajectory from physical interaction, we use a full kinesthetic trajectory

demonstration by the human after each task execution to update the sum of the features over the trajectory (3). This results in the following incremental update rule:

$$\boldsymbol{\theta}_H^{i+1} = \boldsymbol{\theta}_H^i + \alpha\Big(\boldsymbol{\Phi}\Big(\boldsymbol{\xi}_H^i; \mathcal{C}\Big) - \boldsymbol{\Phi}\Big(\boldsymbol{\xi}_R^i; \mathcal{C}\Big)\Big), \tag{5}$$

at iteration i, with learning rate $\alpha \in (0, 1]$. Intuitively, the update rule is a gradient that shifts the weights in the direction of the human's observed feature count. It should be noted that we update the path preferences only using the position part of the state, and the velocity preferences are updated depending on where in space the velocities were observed.

2.2. Features and Rewards

We define the objective function for trajectory optimization as a combination of human rewards and robot objectives. The human rewards consist of features that capture human preferences (1)–(2), whereas the robot objectives define a basic behavior for the robot. Moreover, the robot objectives counterbalance the effect of the human rewards in the optimization, while we learn the weights in the human rewards (Section 2.1). The weights in the robot objectives are hand-tuned. In this section, we first describe the features associated with the human rewards, and then the robot objectives.

The human preferences are captured via the four features listed in Sections 2.2.1–2.2.4 (see Figure 1 for an example of the listed path preferences). We chose these features as they characterize dominant behaviors in manipulation applications that depend on user preferences. Additionally, the features cover the different dimensions of the workspace (in space and time), creating a complete definition of motion behavior. The robot's objectives are composed of the rewards listed in Sections 2.2.5–2.2.7.

2.2.1. Height from the Table

The preferred height from the table, in the range of "low" to "high" is captured by the sigmoid function $\phi_h = \frac{1}{1+e^{-\lambda(h+p)}}$, with h indicating the vertical distance from the table, p indicating the center of the function (an arbitrary "medium" height above the table) and λ indicating the parameter defining the shape of the function. The choice of a sigmoid function is to hinder the effect of this preference when close to upper and lower boundaries during the weight update (e.g., a demonstration at 75 cm above the table should not impact the weight update very differently from a demonstration at 70 cm). The decreasing slope at the boundaries additionally allows other objectives to have a higher impact on the trajectory in such regions during optimization.

2.2.2. Distance to the Obstacle

We encode the user's preferred distance to the obstacle, in the range of "close" to "far", using the exponential feature $\phi_d = e^{-\beta d^2}$, where d is the Euclidean distance to the center of the obstacle and β is the shape parameter. This exponential function gradually drops to 0 at a certain distance from the obstacle. This distance is a threshold outside which the local behavior of the optimization is no longer affected by the distance to the obstacle. Importantly, if a negative weight is learned associated with this feature, the trajectory is still attracted towards the obstacle even if the initial trajectory lies outside of this threshold. This is because our optimization strategy globally explores different regions of the workspace, and in this case it would detect that there is a reward associated with being closer to the obstacle.

2.2.3. Obstacle Side

We define this feature in the range of "close" (the side of the obstacle closer to the robot) to "far" (the side of the obstacle far from the robot) via the tangent hyperbolic function $\phi_s = \frac{2}{1+e^{\gamma S}} - 1$. Here, S is the lateral distance between a trajectory sample and the vertical plane at the center of the obstacle and γ is a shape parameter. This symmetric function is designed to have a large span in order to be active in all regions of the workspace. However,

as the gradient of this function decreases at larger lateral distances, so does the influence of this function on the local trajectory optimization.

2.2.4. Velocity

To encapsulate the user's velocity preferences, we adopt a different approach using a discretized linear combination of uniformly distributed Radial Basis Functions (RBFs) in the range $[\dot{x}_{\min}, \dot{x}_{\max}]$. For each segment r, we map the average velocity norm $\bar{\dot{x}}_r$ onto these RBFs, given by:

$$\psi_j(\bar{\dot{x}}_r) = e^{-\left(\varepsilon \bar{\dot{x}}_r - c_j\right)^2}, \tag{6}$$

where the shape variable ε defines the width and c_j defines the center of the jth RBF, with $j = 1, 2, \ldots, n$ (we use $n = 9$).

Inspired by Fahad et al. [22], we discretize the above feature to two bins, based on the distance d_r of each segment center to the obstacle. Hence, we have two cumulative feature vectors: $\mathbf{\Phi}_{V1}$ for $d_r \in [0, d_c)$ and $\mathbf{\Phi}_{V2}$ for $d_r \in [d_c, \infty)$. This allows us to approximate the speed of motion separately in areas considered to be "close" to or "far" from the obstacle based on the distance threshold d_c (obtained from demonstration data). This way, we capture velocity preferences relative to the obstacle position. Similarly, features can be defined relative to other context parameters to capture velocity preferences that depend on other parameterized positions.

However, the issue might arise that the two trajectories do not have the same number of segments in each distance bin. In such a case, we employ feature imputation using the mean of the available values.

The following subsections describe the rewards that make up the robot's objectives.

2.2.5. Path Efficiency Reward

We calculate the total length of a trajectory, which we use as a negative reward. Penalizing the trajectory length is essential in counterbalancing the human preference features in the optimization process. Essentially, it pulls the trajectories towards the straight line path from start to goal and rewards, keeping them short.

2.2.6. Collision Avoidance Reward

We use the obstacle cost as formulated by Zucker et al. [23], which increases exponentially once the distance to the obstacle drops below a threshold. The negative cost is our reward.

2.2.7. Robot Velocity Reward

This reward achieves a low and safe velocity in the absence of human velocity preferences and is defined based on (6). In IRL, it is beneficial to learn how people balance other features against a default reward [24].

2.3. Motion Planning via Trajectory Optimization

We discuss the problem of motion planning in two parts. First, we address the optimization of the path of the trajectory in the workspace. We then address the optimization of the velocity along this path, defining the timing of the motion.

Solving the path optimization problem over the Cartesian task space would be complex and inefficient. Instead, we employ a trajectory planning algorithm [25] that interpolates between waypoints with piecewise clothoid curves. This algorithm minimizes the acceleration, which results in a smooth and realistic motion. We exploit this algorithm to significantly reduce the search space for the path optimization and sample trajectories using a vector of waypoint coordinates $\mathbf{p}$ and its corresponding time vector $\mathbf{t}_P$, $\boldsymbol{\xi} = f(\mathbf{p}, \mathbf{t}_P)$.

We consider three waypoints $\mathbf{p} = [\mathbf{p}^s; \mathbf{p}^m; \mathbf{p}^g]$, corresponding to the start position, an arbitrary position within the path and the goal position, respectively. We further simplify the problem by fixing the time vector to $\mathbf{t}_P = t^g[0; \frac{D(\mathbf{p}^m)}{D(\mathbf{p}^g)}; 1]^T$, where $D(\cdot)$ indicates the

Euclidean distance of a waypoint to $\mathbf{p}^s$ and t^g is the time, we assume all trajectories take to finish, just for the path optimization (the shape of the paths is not affected by t^g in the time ranges of our manipulations; therefore, we assume the path to be independent of velocity). An uneven distribution of waypoints would bias the reward value. Setting up the time vector in this manner ensures a constant velocity throughout the trajectory, which results in an even distribution of samples over the path. Trajectories can then be sampled only as a function of waypoint positions $\boldsymbol{\xi} = f(\mathbf{p})$.

We then solve for the optimal waypoint vector $\mathbf{p}^*$ using the following non-linear program formulation:

$$\begin{aligned} \mathbf{p}^* &= \arg\max_{\mathbf{p}} \left(R_P(\mathbf{p}; \mathcal{C}, \boldsymbol{\theta}_{HP}) + \boldsymbol{\theta}_{RP}^T \boldsymbol{\Phi}_{RP}(\mathbf{p}; \mathcal{C}) \right), \\ &\text{subject to: } \boldsymbol{h}(\mathbf{p}) = \mathbf{0},\ \mathbf{p}_{\text{low}} \leq \mathbf{p} \leq \mathbf{p}_{\text{upp}}. \end{aligned} \tag{7}$$

Here, the objective function consists of the human path reward R_P and the robot's path objective, which is a linear combination of predetermined weights $\boldsymbol{\theta}_{RP}$ and the aforementioned path reward functions $\boldsymbol{\Phi}_{RP}$. The equality constraint ensures the start and goal positions are met. As a result, we are effectively searching for the waypoint $\mathbf{p}^m$ that maximizes the objective function. The upper and lower boundaries $\mathbf{p}_{\text{low}}$ and $\mathbf{p}_{\text{upp}}$ limit the trajectory to stay within the robot's workspace. Once $\mathbf{p}^*$ is found, we construct the full trajectory using $\boldsymbol{\xi}_P^* = f(\mathbf{p}^*, \mathbf{t}_P)$. Figure 3 shows an example of the convergence of the optimizer towards a path that adheres to "low height", "close side" and "close to obstacle" preferences.

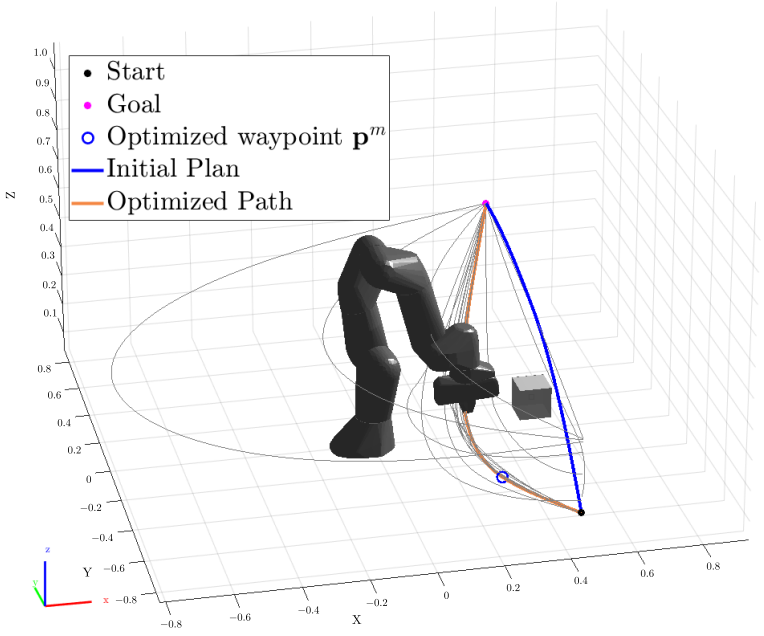

Figure 3. An example of convergence towards the optimal path. The optimizer places $\mathbf{p}^m$ in different locations in the workspace to generate different paths. The paths explored by the optimizer are indicated in gray. The orange path indicates the output of the path optimizer, resulting from placing the middle waypoint at the location indicated by the blue circle.

Having the optimal path $\boldsymbol{\xi}_P^*$, we divide the trajectory into M segments (as described in Section 2.1). Next, we store the positions of the waypoints at the end of the segments in $\mathbf{p}_V^* = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_M]$. This vector is fixed to maintain the shape of the trajectory. The corresponding timestamps, stored in $\mathbf{t} = [t_1, t_2, \ldots, t_M]$, are the variables we optimize. Thus, trajectories sampled by the optimizer are only a function of the time vector $\boldsymbol{\xi} = f(\mathbf{t})$.

By optimizing $\mathbf{t}$, we optimize the average velocity of each segment. The optimal time vector is

$$\begin{aligned} \mathbf{t}^* = \arg\max_{\mathbf{t}} \Big(R_V(\mathbf{t}; \mathcal{C}, \boldsymbol{\theta}_{HV}) + \theta_{RV} \phi_{RV}(\mathbf{t}; \mathcal{C}) \Big), \\ \text{subject to: } \boldsymbol{g}(\mathbf{t}) \leq \mathbf{0},\ \mathbf{t} \leq \mathbf{t}_{\text{upp}}, \end{aligned} \tag{8}$$

where the objective function is composed of R_V and the robot's velocity objective ϕ_{RV}, which provides a reward for carrying objects at $\dot{x}_{\text{robot}}$ with a fixed weight θ_{RV}. The inequality constraint $\boldsymbol{g}(\mathbf{t})$ bounds the velocity over each segment to $\dot{x}_{\min}$ and $\dot{x}_{\max}$, not allowing the timestamps to get too close or far from each other. The upper boundary on $\mathbf{t}$ acts as a limit on the total duration of motion.

Finally, the trajectory that adheres to both the path and velocity preferences is constructed using $\boldsymbol{\xi}_R = f(\mathbf{p}_V^*, \mathbf{t}^*)$. The full method is summarized in Algorithm 1.

Algorithm 1: Learning human preferences from kinesthetic demonstration

Record $\boldsymbol{\xi}_H^0 = \{\boldsymbol{x}_k, t_k\}_{k=1}^N$, obtain context $\mathcal{C}$
$\dot{\boldsymbol{x}}_k \leftarrow \frac{d}{dt}\boldsymbol{x}_k$, compute $\bar{\dot{x}}_r$ and $\bar{\boldsymbol{x}}_r$
Initialize $\boldsymbol{\theta}_H^0, \boldsymbol{\theta}_R, \boldsymbol{\xi}_R^0$
Set $i = 0$
while *executing task* **do**
  if *Received Human Feedback* **then**
    $\boldsymbol{\theta}_H^{i+1} = \boldsymbol{\theta}_H^i + \alpha\Big(\boldsymbol{\Phi}\Big(\boldsymbol{\xi}_H^i; \mathcal{C}\Big) - \boldsymbol{\Phi}\Big(\boldsymbol{\xi}_R^i; \mathcal{C}\Big)\Big)$
  $\mathbf{p}^* \leftarrow \text{Optimize}(\boldsymbol{\theta}_{HP}^{i+1}, \boldsymbol{\theta}_{RP}, \mathcal{C})$
  $\mathbf{t}^* \leftarrow \text{Optimize}(\mathbf{p}^*, \boldsymbol{\theta}_{HV}^{i+1}, \theta_{RV}, \mathcal{C})$
  $\boldsymbol{\xi}_R = f(\mathbf{p}^*, \mathbf{t}^*)$
  $\boldsymbol{\tau} \leftarrow \text{Impedance}(\boldsymbol{\xi}_R)$
  $i = i + 1$

3. Method Validation with User Study

To validate our framework, we conducted two user experiments on a Franka Emika 7-DoF robot arm. Thereby, we demonstrated a proof-of-concept of our approach in a real-world scenario with non-expert users. In both experiments, we use a set of three pick-and-place tasks in an agricultural setting, as shown in Figure 4. The primary goal of each task was moving the tomatoes from the initial position to the goal without any collisions with the obstacle. The experiments were approved by the Human Research Ethics Committee at Delft University of Technology on 6 September 2021.

We recruited 14 participants (4 women and 10 men) between 23 and 36 years old (mean = 26.8, SD = 3.6), 6 of whom had prior experience with robotic manipulators, but none of whom had any exposure to our framework.

Each user first took approximately 10 min to get familiar with physically manipulating the robot in the workspace. In this period, we also instructed users about the goal of the task and the preferences the robot could capture. Users then proceeded with the two experiments. To subjectively assess whether the framework can capture a range of different behaviors, in the first experiment, we let the users freely choose their path and velocity preferences. Once users were more familiar with the framework, in the second experiment, we assessed how effectively they could teach a set of pre-defined preferences to the robot. The overview of the user study is provided in Figure 5. We discuss each experiment in the following subsections. A video of the experiments can be found here: https://youtu.be/hhL5-Lpzj4M, accessed on 13 April 2023.

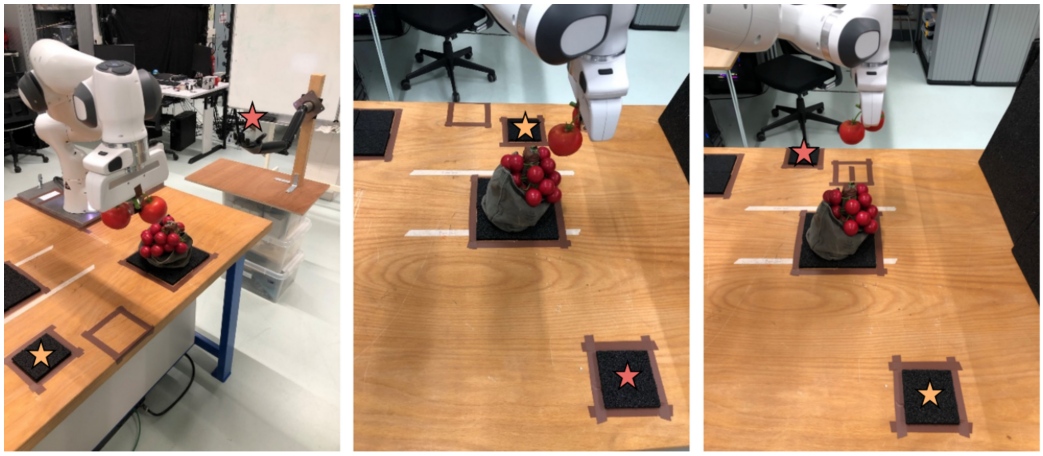

Figure 4. From left to right: Scenarios 1–3. The orange and red star, respectively, indicate the start and goal positions. The obstacle to be avoided is the bag of tomatoes. Scenario 1 and 2 shared the same starting positions, and Scenario 2 and 3 shared the same obstacle positions. Notice the difference in height of the goal position in Scenario 1 compared to Scenarios 2 and 3.

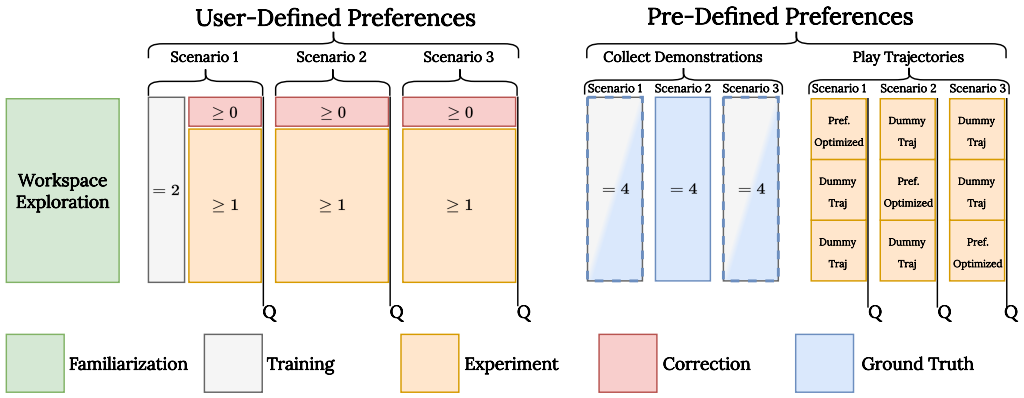

Figure 5. The experimental protocol. Users started with workspace familiarization, then went through the first experiment assessing the performance of the framework in understanding their preferences. Finally, in the last experiment, they provided ground truth demonstrations and evaluated the demonstrated trajectories in adhering to the set of predefined preferences. The numbers indicate the number of demonstrations given, either by the human (training/correction/ground truth) or the robot (experiment). The order in which the dummy trajectories were shown to the users was different in every scenario. The symbol "Q" indicates when participants were provided with questionnaires.

3.1. User-Defined Preferences

In the first experiment, we investigated how our framework performs when users openly chose their set of preferences. We were specifically interested in assessing how well the robot plans motions in new task instances with a context it has not seen before (i.e., generalization of preferences to new scenarios). We also evaluated the user experience in terms of acceptability and effort required from the user's perspective. Accordingly, we tested the following hypotheses:

Hypothsis 1 (H1). *The proposed framework can capture and generalize user preferences to new task instances.*

Hypothsis 2 (H2). *Users feel a low level of interaction effort.*

3.1.1. Procedure and Measures

Users first performed a demonstration in Scenario 1 (Figure 4) for path preferences with the robot in gravity compensation mode. Notably, we did not limit users to a discrete set of preferences. For instance, instead of asking users to pass on either the close or far side of the obstacle, we asked them to intuitively demonstrate how far to either side of the obstacle they would prefer to pass. They could, for example, decide to pass right above the obstacle, which would correspond to a "stay to the middle of the obstacle" for the "obstacle side" path preference. We then collected a second separate demonstration for the velocity preferences. During velocity demonstrations, the robot was only compliant along a straight line path covering the full range of distances to the obstacle. This allowed the users to demonstrate their preferred speed without having to care about the path. The velocity optimization step can take up to 3 min; therefore, we simplified the method for learning and planning velocity preferences to find the velocity c_j with the highest feature count in this part of the study. Users were instructed to provide corrections via additional kinesthetic demonstrations (max 10 min per scenario) until they were satisfied with the resulting trajectory. However, the users were informed that the trajectory speed was only trained once and would not be updated further.

After observing each trajectory, the users filled out a subjective questionnaire for qualitative evaluation, rating the following statements on a 7-point Likert scale:

1. The robot accomplished the task well.
2. The robot understood my **path** preferences.
3. The robot understood my **motion** preferences.

To evaluate the effort, we counted the number of times a user provided feedback, and let the participants fill out the NASA Task Load Index (TLX) at the end of this experiment. The independent variables of this experiment are the contexts which are varied for each scenario for assessing workload. Although we do not compare results with a baseline here, NASA-TLX is still appropriate since it can capture absolute results [26].

3.1.2. Results

Users demonstrated a multitude of path preferences, including "keep a low distance to the obstacle" and "stay at a medium height above the table". Similarly, for velocity preferences, while the majority opted for a constant "medium" speed, both the preferences of going "slower when close to the obstacle" and "faster when close to the obstacle" were demonstrated at least once.

Figure 6A shows that the average amount of feedback given to the system after the first task dropped, with the majority of the users satisfied with the results of generalization after the initial demonstration (we counted the training step in Task 1 as feedback). This result is also reflected in Figure 6B, showing that the users scored the first trajectory produced in every scenario consistently high for all three statements, supporting the claim that the framework can generalize both path and velocity preferences to new task instances. This provides strong evidence in favor of both **H1** and **H2**.

The NASA-TLX results in Figure 7 show that the users experienced low mental and physical workload. Although kinesthetic teaching is normally associated with high effort, our framework's effort scores remain mostly on the lower side of the scale. One participant was particularly strict on a height preference the algorithm failed to capture, resulting in three iterations of feedback in Scenario 1. Overall, the results in Figure 7 support **H2**.

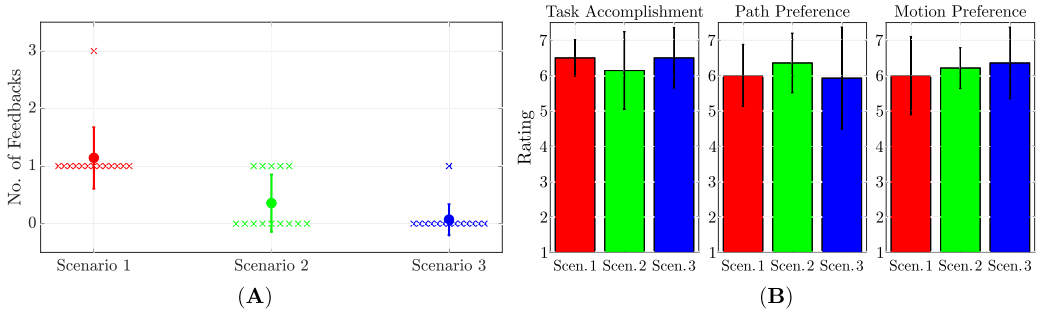

Figure 6. Results of the first experiment. (**A**) Average amount of feedback provided to the system for each task. The dot represents the mean score, the error bars represent the standard deviation, and the crosses indicate individual data points. (**B**) Results of the Likert questionnaire for the first resulting trajectory in every task (i.e., prior to any additional demonstrations). The error bars correspond to standard deviation.

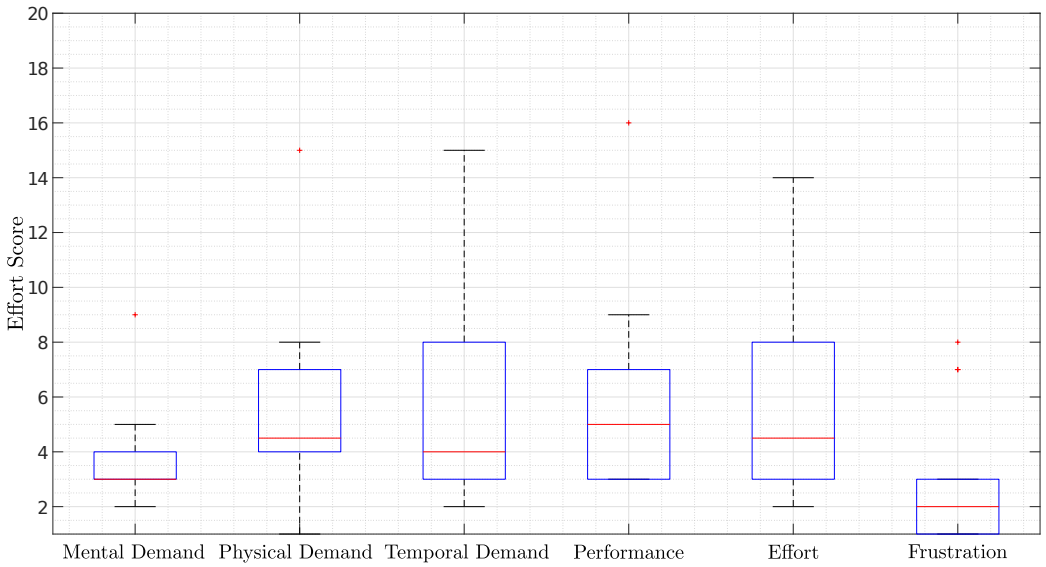

Figure 7. Results of the NASA-TLX questionnaire after the first experiment.

3.2. Pre-Defined Preferences

To objectively evaluate the accuracy and the user's ability to discern preferences, we conducted an experiment where users are asked to adhere to the following path preferences (we did not consider velocity preferences in this experiment):

- Pass on the side of the obstacle that is closer to the robot.
- Stay far from the obstacle.
- Keep a high elevation from the table.

Exactly how to express these preferences and how to trade off between them if necessary was left to the users. We tested the following hypotheses:

Hypothsis 3 (H3). *The method remains consistently accurate in all scenarios.*

Hypothsis 4 (H4). *Users can clearly distinguish that the output of the framework follows the specified preferences.*

3.2.1. Procedure and Measures

We collected four demonstrations per scenario. For half of the participants, we trained the model on the mean of the four demonstrations from the first scenario, and for the other half, we used the mean of data from the third scenario. This was to establish that our method can be generalized, even when changing the set used as the training data.

After that, the users were shown three trajectories per scenario: the output of our framework and two dummy trajectories (Figure 8). The dummy trajectories were designed to adhere to two out of three path preferences. This allowed us to observe if users could distinguish our method's results compared to sub-optimal trajectories.

As an objective measure of the accuracy of our method, we computed, per scenario, the total Euclidean distance of samples within each trajectory with respect to the mean of the demonstrations (using $N = 80$). Furthermore, we compared the total feature counts along each trajectory and measured the error with respect to the ground truth in the feature space.

Subjectively, users rated a 7-point Likert scale per trajectory: "the robot adhered to the demonstrated preferences".

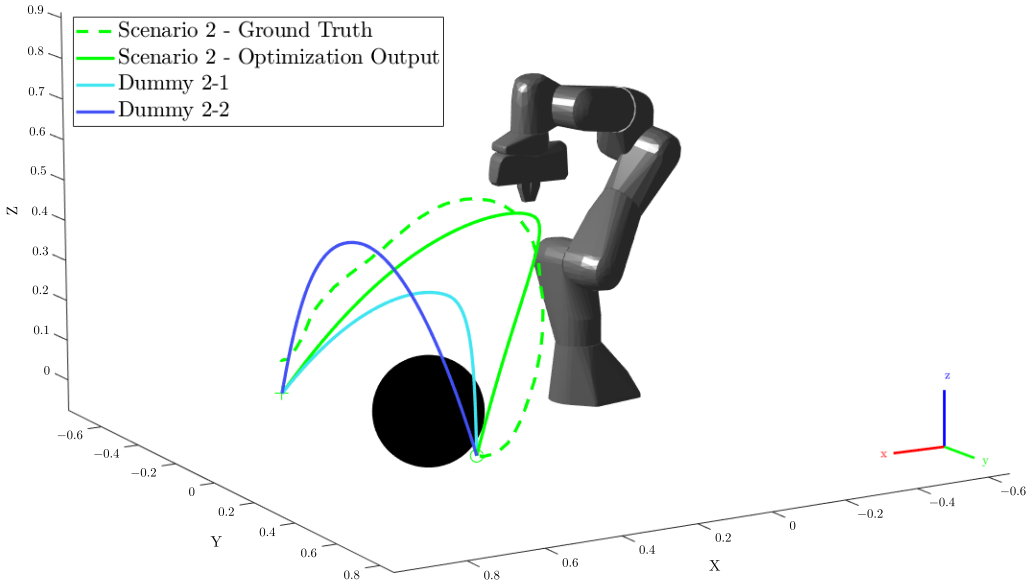

Figure 8. Scenario 2 results (second experiment) for a single user. The dummy trajectories, in light and dark blue, are designed not to meet the "height from table" and "obstacle side" preferences, respectively. The green dashed and solid lines are the mean of human ground truth demonstrations and the robot trajectory, respectively. The black sphere represents the obstacle. The framework was trained on data from Scenario 3 and had no access to the ground truth shown.

3.2.2. Results

Figure 8 shows a generalization of the results of our method under the aforementioned path preferences. The robot attempted to capture and optimize each user's personal interpretation of the preferences (e.g., one user's definition of "high" is different from another). We show the combined results of all users in Table 1, listing the trajectories' mean, min and max Euclidean distance to the ground truth, normalized relative to the start-to-goal distance in each scenario (respectively 1.08, 0.74 and 0.88 m). The optimized trajectories have the smallest error, but the results only partially support **H3**, as the errors in Scenarios 2 and 3 are slightly larger than in Scenario 1. This scenario has the longest distance from start to goal, for which the framework seems to perform better.

Table 1. Average distance error of trajectory samples with respect to the ground truth, normalized with respect to the distance from the start to the goal in meters: mean [min, max].

	Scenario 1	**Scenario 2**	**Scenario 3**
Optimized	0.14 [0.09, 0.18]	0.20 [0.12, 0.27]	0.17 [0.13, 0.24]
Dummy 1	0.24 [0.13, 0.34]	0.26 [0.16, 0.38]	0.30 [0.21, 0.41]
Dummy 2	0.27 [0.18, 0.33]	0.39 [0.28, 0.47]	0.23 [0.18, 0.28]

Figure 9 shows the errors of the trajectories in feature space. In all scenarios, our optimization result occupies the smallest area. However, in Scenarios 2 and 3, the optimized trajectories occupy a slightly larger area than in Scenario 1, showing the same trend of performance loss in scenarios with a shorter length. Furthermore, in Scenarios 2 and 3, dummy trajectories occasionally perform slightly better for one of the preferences. Nevertheless, we see in Figure 10 that users clearly score the output of our framework higher, which strongly supports **H4**. This indicates that users prefer all preferences to be satisfied simultaneously. The best performing dummy (S3-D2), with the smallest area in Figure 9 and lowest values in Table 1, correlates to a high rating in Figure 10. This also supports **H4**, suggesting that non-expert users intuitively recognize such preferences in trajectories.

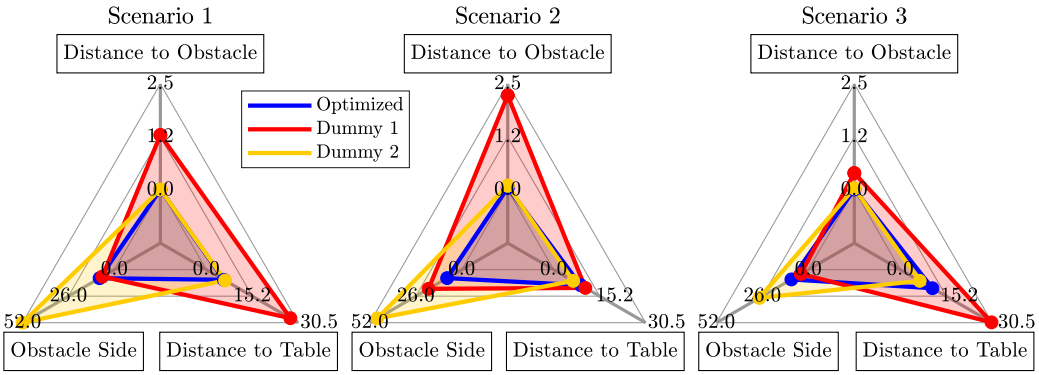

Figure 9. Total feature count errors of each path preference (all participants) with respect to the ground truth (i.e., smaller values for each axis are favored).

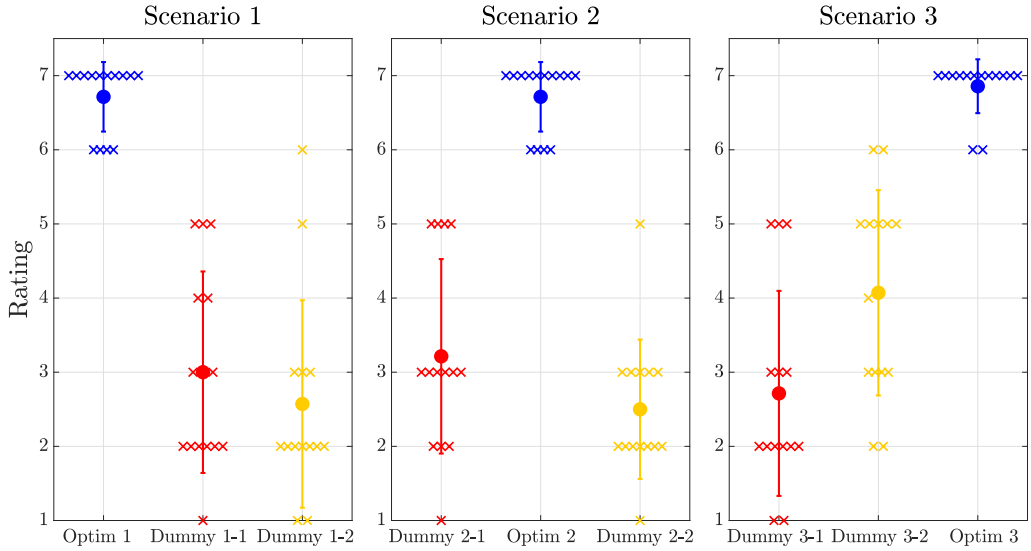

Figure 10. Result of Likert questionnaire for experiment 2. Crosses indicate individual ratings, while the dots and error bars, respectively, represent the mean and standard deviation. Users clearly recognize and highly rate the output of the framework in terms of adhering to path preferences.

3.3. Discussion

As the state-of-the-art methods do not have the same functionalities (e.g., path–velocity separation) as the proposed method, we conducted a user study only on the proposed method itself. To account for that, we employed absolute types of metrics (i.e., Likert and NASA-TLX), which can be interpreted independently, rather than tied to a specific external baseline. For example, the Likert scale is tied to an agreement with the given statements and the natural point on the agreement scale serves as a general baseline. The advantage of this is that the results are not tied to a specific relative baseline. If methods that enable the

same functionalities are developed in the future, the same Likert scale/questionnaire can be employed to compare the subjective results independently of a specific baseline.

An advantage of the proposed method is that it learns fast. During the first part of the user study, participants spent on average 16.5 s interacting with the robot before expressing satisfaction with the results. This is partially due to having access to kinesthetic demonstrations. This method of demonstration has been criticized as challenging in applications involving high DoF manipulators [1,27]. However, the separation of learning and control in our framework means that users do not have to provide the correct configuration of the arm in their demonstrations. This feature made it significantly easier for the users to provide demonstrations, which is reflected in the reported low mental and physical loads (Figure 7).

The separation of path and velocity planning has additional benefits. Formulating the optimization as a multi-objective problem with both position and velocity features results in undesirable interactions of objectives. For instance, when velocity features reward high speeds, the trajectory converges to a longer path. Conversely, path features with high rewards in specific regions of space result in slow motion in those regions to increase the density of samples and consequently the overall reward. On the other hand, the separated trajectory optimization has the limitation that it cannot account for dynamical quantities such as joint velocity and acceleration, and the efficiency of movements in the robot's joint space cannot be considered.

A challenge with our definition of robot and user objectives is that the trajectory optimization outcome does not always align with task requirements. For instance, a strong "stay close to the object" preference can result in a minimum cost for a trajectory that is briefly in a collision. Tuning the collision weight can only partially solve this issue, as at a certain point, this cost can interfere with the path preferences.

Our user study results showed that non-expert users can intuitively use our method to quickly teach a wide range of preferences to the robot. Although the generalization results of different task instances show that we do not always reproduce trajectories with the exact desired shapes in the workspace (see Figure 8), the subjective performance evaluation shows that users still deem these trajectories highly suitable in terms of task accomplishment and the preferences achieved. State-of-the-art LfD methods are very capable of producing accurate and complex dynamic movements [28]. However, in tasks where there are multiple ways of achieving the same goal, we prefer to trade off motion accuracy for achieving planning propensities on a higher level.

Unfortunately, our approach inherits the limitations of IRL approaches that require specifying reward features by hand. Both features and robot rewards depend on several parameters which require tuning. The problem becomes especially difficult as our features simultaneously govern the behavior of reward learning and trajectory optimization. For instance, high gradients in the feature function lead to erratic behavior of the optimizer, leading to poor solutions and convergence to local optima. Yet, for certain features, a sufficiently high gradient is required to facilitate the learning of preference weights that are large enough to counterbalance each other. As a result, we had to resort to further tuning of parameters, such as the learning rate in (5). An interesting direction for future work would be to test whether and how well these issues can be alleviated by feature learning from additional demonstrations, as was demonstrated by Bobu et al. [29]. Furthermore, an approach similar to that in [30] could be employed to learn the relative weighting among features and add additional features through nonlinear functions using neural networks.

In feature engineering or learning, the definition of the context determines how expressive the features are. We considered a limited set of vectors as the context in this work (i.e., obstacle position and start and goal positions). It is possible to include additional information, such as object properties (e.g., sharp, fragile or liquid) [1], human position [4,6] and the number of objects. The more rich the context, the more preferences the model can capture in complex environments. However, training diversity can become an issue with contextually rich features, as the model would require more demonstrations to cover

a wider range of situations. This will increase the training time. An evaluation of the trade-off between improved generalization and higher training time is left for future work.

4. Supplementary Comparison Study

The purpose of this supplementary study is to highlight different aspects and properties of our method in comparison to two common methods from the literature: Dynamic Movement Primitives (DMPs) [31] modified with potential fields for obstacle avoidance [9], and the method used by Bajcsy et al. [4] (referred to as PHI). Since these methods are different conceptually and by design (i.e., optimized for different properties), a quantitative comparison is not meaningful. Thus, we examined their aspects in a practical transportation task qualitatively. These aspects are adherence to preferences, robot objectives, trajectory feasibility and online learning. In the following subsections, we first discuss the different aspects in more detail, before showing the effects in the transportation task and discussing the pros and cons of the different methods.

4.1. Conceptual Differences per Aspect

4.1.1. Adherence to Preferences

The methods capture preferences in a different way. Even though we added obstacle awareness to the DMPs we compared to, they lack an explicit notion of preferences. A forcing function was learned to match the shape and velocity distribution of the demonstration, but without any parameterization over features that may capture behavior relative to the context. The potential fields for obstacle avoidance add a basic level of context awareness, but a predefined one.

Both our method and PHI learn an explicit preference model that is structured as a linear combination of context-parameterized features. Similar to our model, PHI considers the "*height from the table*" and "*distance to the obstacle*". We additionally consider the "*obstacle side*", such that our features cover the different dimensions in space and allow us to capture the preferences in every direction. PHI instead considers other features, such as "*distance to human*" and "*efficiency*".

Our features are counterbalanced by explicit robot objectives (Section 4.1.2). In PHI, it is possible to replace the features with the features we use, including the ones for the robot which will not be updated during learning. This way, we can test the effects of the change in features and the change in method.

4.1.2. Robot Objectives

In contrast to PHI, we chose to explicitly separate objectives such as "*path efficiency*" and "*collision avoidance*", from the preferences we tried to capture. Instead, we let the robot have a reward function of its own. The same effect can be achieved in PHI by fixing the weights of selected features.

The effects of the trade-off between the learned human rewards and the given robot objectives visible in the iterative updates can be viewed as a negotiation between the preferences of two independent agents. We believe that this separation and negotiation will be beneficial, especially as tasks become more complex and the artificial agent has knowledge complementary to the human. The benefits may be less visible in the simple task considered in this paper.

As DMPs do not explicitly model an objective function to be optimized, this attribute does not apply.

4.1.3. Trajectory Feasibility

Our method does not automatically check if the planned trajectory is feasible to execute by the robot. A motion feasibility objective can be added to the robot objective function to take this into account in the path optimization.

Rather than weighing the learned preferences against robot objectives, PHI ensures motion feasibility by optimizing the trajectory in the robot configuration space. This re-

quires an additional simulation step, incorporating the kinematic model of the robot during trajectory optimization. However, no corrections then need to be applied in hindsight to ensure trajectory feasibility.

4.1.4. Online Learning

Our method requires a full trajectory to learn from, whereas PHI updates the internal model at each time step. This potentially makes our method less efficient. On the other hand, it allows us to capture velocity preferences in addition to path preferences. Furthermore, because we separate the demonstration from the execution, we obtain a more "clean" observation of the preferences, as we do not have to deduce from the interaction forces what the human demonstration would have looked like without the robot interference. This is likely to benefit generalization. In the case of large user corrections, it may even reduce the user's effort to demonstrate their preferences without the robot interfering. Velocity preferences have been found to be especially cumbersome to correct in an online manner [5].

When it comes to online updates, DMPs are the fastest because there is no complex underlying model that needs to be updated. However, the trade-off of having a much simpler model is that it lacks the ability to capture preferences in a way that might generalize to changes in the scenario.

All three methods update their model to reduce the error with respect to the latest observation from demonstration. A learning rate trades off learning and overfitting on the corrective demonstration. DMP updates correct behavior on the trajectory level, whereas PHI and our method update at a higher level, where the observed trajectories are considered a consequence of a human reward model. Nevertheless, future observations that appear contradictory to earlier ones will cause (partial) unlearning of the earlier updates. This results in erroneous behavior learned from imperfect corrections to be corrected, but may in some cases also lead to undesired unlearning.

4.2. Comparison

We will now present a qualitative comparison between the three methods, PHI with two different feature sets: $\_\phi_{\text{orig}}$, $\_\phi_{\text{our}}$. Our aim is to show the effects of the conceptual differences discussed in Section 4.1. To make the comparison as fair as possible, we let all the models learn from the same demonstration data. All methods have access to and consider the obstacle position for planning.

We modified PHI to bypass the estimation of the human-desired trajectory from forces, as we have direct access to the desired trajectory from demonstration. We computed the "human correction" every time step from the mismatch between the planned trajectory and the demonstration. The trajectory optimization in PHI requires a robot model for the optimization. As our trajectory optimization does not take robot dynamics into account, we used a fully actuated point mass for the trajectory optimization. In order to achieve comparable smooth optimal paths, we interpolated the trajectory with a spline instead of linearly as was done originally. For both sets of features, the feature weight ranges and update rates were hand-tuned to achieve as close a trajectory match in the initial scenario as we could manage. This initial scenario is illustrated in Figure 11.

We consider a situation where the user has a preference for "passing on the close side of the obstacle" due to the existence of a wall on the other side that the robot is not aware of. Furthermore, we want to "remain close to the obstacle" and to "slow down when passing close to the obstacle". We use a single kinesthetic demonstration containing these three preferences as the input to all methods. For PHI_ϕ_{orig}, we obtained the correct choice of obstacle side in Figure 11 by assuming a person standing on the other side of the obstacle and making use of their "human feature", learning not to come too close to the human.

Figure 11 shows the demonstration we used for training, as well as the trajectories obtained from the three methods. As the results are generated for the same context as in the demonstration, these results reflect the performance prior to any generalization of

preferences. As PHI updates its internal model at every time step, we observed partial unlearning of some features towards the end of the trajectory. This is particularly visible for the "obstacle distance" in PHI_ϕ_{orig}. In Figure 11, we show an additional trajectory PHI_$\phi_{\text{orig},\tau=0.45}$, which was generated by PHI with the original features and the weights learned at 45% of the trajectory. We see that PHI_$\phi_{\text{orig},\tau=0.45}$ is considerably closer to the demonstrated trajectory. The demonstrated trajectory has many waypoints close together and quite close to the obstacle, as it slows down when passing it. PHI, on the other hand, has equally spaced waypoints. As a result, towards the end of the trajectory, a considerable batch of PHI waypoints is further away from the obstacle by default. When the weights continue to update on the difference, we obtain the trajectory PHI_ϕ_{orig}, which lies closer to the obstacle. With our features, in PHI_ϕ_{our}, the effect is less pronounced as the features trade off differently, yet the learned path is still different from our trajectory, as PHI uses a different trajectory optimization method.

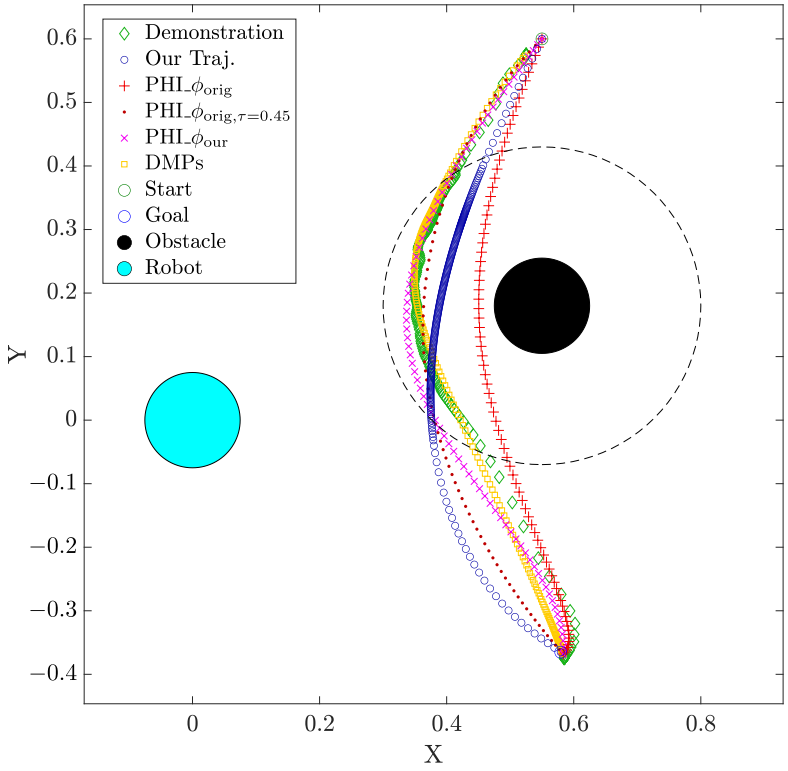

Figure 11. Training scenario with the human demonstrated trajectory (green diamonds) and the learned reproductions: ours is represented by dark blue circles; PHI_ϕ_{orig} is represented by red plus signs, with the intermediate learning result represented by dots; PHI_ϕ_{ours} is represented by purple crosses; and DMP is represented by yellow squares. By placing the markers at equal time intervals, we display the velocity of the trajectories (i.e., the closer the markers, the slower the motion). As PHI does not support differences in velocity, all red and purple markers are spaced equally along the trajectory. The black, cyan, blue and green circles, respectively, represent the obstacle, robot, goal (bottom) and start (top) positions. For this study, we set d_c = 22.5 cm (indicated by the dashed circle). We consider points within this region as "close" to the obstacle.

Especially considering PHI_ϕ_{our}, all three methods perform reasonably well in terms of adhering to the aforementioned path preferences, with a slight variation in how close the robot passes by the obstacle. As discussed in Section 4.1, PHI is not able to capture any velocity preferences. Notably, DMP performs well in this aspect, as it is able to replicate the demonstrated behavior in terms of both path and velocity.

Next, we modified the scenario nine times in three different ways, changing the goal, start and obstacle positions. We compared how each method was able to generalize the

initial observation to the different contexts. Figure 12 displays the trajectories produced by the three methods in the nine new scenarios, where PHI again has two different feature sets.

We observe that the trajectory by our method (shown in dark blue) passes on the left side of the obstacle close to the robot in every case, and the velocity preference of slowing down when passing close to the obstacle is only achieved by our framework.

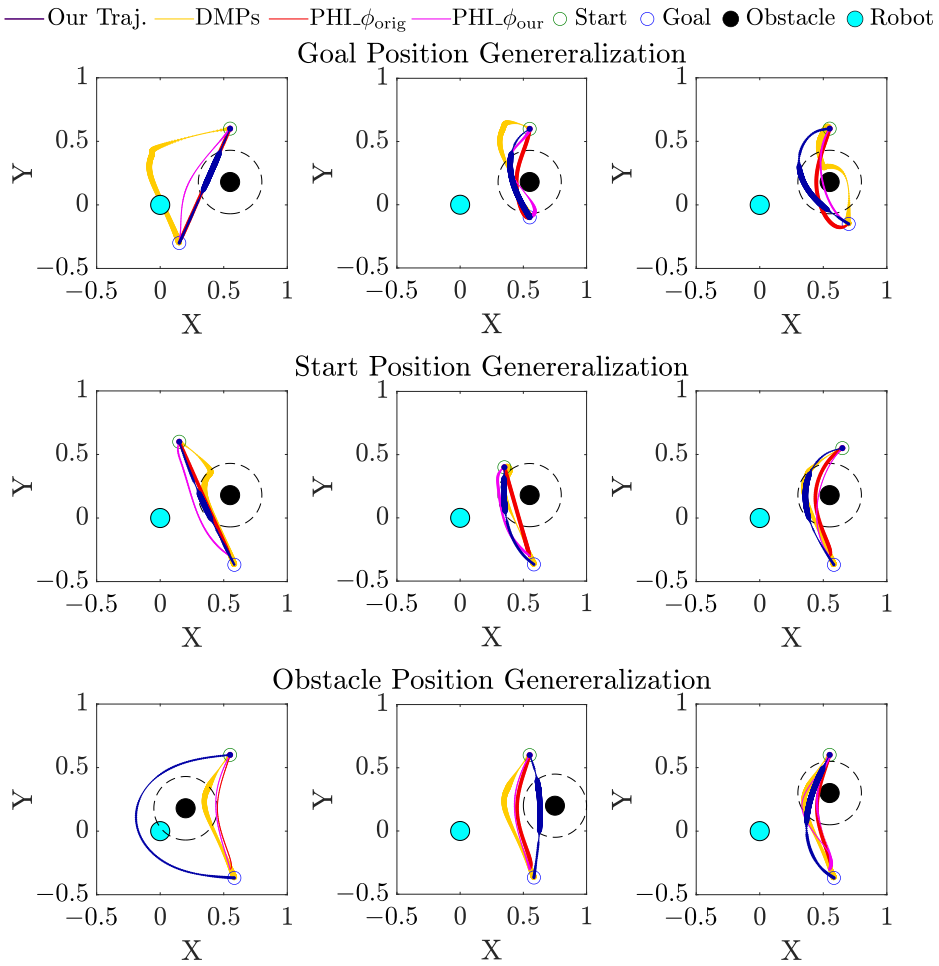

Figure 12. We demonstrate the generalization of our method by modifying the goal (**top**), start (**middle**) and obstacle (**bottom**) positions. The yellow, blue and red and purple trajectories correspond, respectively, to the output of the DMPs, our framework and the two versions of PHI. The thickness of the line indicates the inverse of normalized velocity (i.e., the thicker the line, the slower the trajectory).

4.3. Discussion

A comparison with DMP illustrates how a lack of a deeper level of knowledge about why a trajectory was demonstrated in a specific manner leads to failure in generalization to new contexts. These results emphasize the need for consideration of human models, such as our reward in (1)–(2) in LfD methods. PHI, with its model, performs considerably better. However, we observe that the internal trajectory optimization reacts differently to the different sets of features, resulting in slight differences in generalized trajectories. The main point, regardless of the applied features, remains that PHI is not able to capture velocity preferences. Table 2 summarizes the strengths and weaknesses of the three methods with respect to the aforementioned aspects.

Table 2. Qualitative evaluation of the different aspects of the three methods: DMPs, PHI and ours. The marker "o" indicates a value between " −" and "+".

	Adherence to Preferences [a]	Robot Objectives [b]	Trajectory Feasibility [c]	Online Learning [d]
DMPs	−	−	−	+
PHI	o	o	+	o
Ours	+	+	o	−

[a] Based on Figure 12, "−" indicates adherence to only a few or inconsistent adherence to many preferences and "+" indicates adherence to most preferences in most of the cases. [b] Based on the model structure, "−" indicates no robot objectives can be added and "+" indicates when arbitrary robot objectives can be added. [c] "−" indicates no guarantees for trajectory feasibility, and "+" indicates trajectory feasibility can be guaranteed at all times. [d] "−" indicates the inability to learn in real-time and "+" indicates the ability to learn and re-plan while the task is being executed.

PHI optimizes the trajectory in the joint space, which can be executed quickly since inverse kinematics is only required at waypoints. It ensures the planned trajectories are feasible for the robot, which can be interpreted as implicit robot objectives being satisfied. On the other hand, our method optimizes the trajectory in the task space; thus, additional inverse kinematics computations are necessary together with an explicit description of corresponding robot objectives. The use of inverse kinematics can also be problematic when there are redundant DoFs or when there are potential self-collisions. Nevertheless, planning in the task space is closer to where the human preferences typically are (i.e., more intuitive) and can handle obstacle avoidance in a manner that is more predictable for a non-expert human.

It should be noted that our framework does take up to two minutes of optimization (total for path and velocity), whereas the DMPs trajectory is produced instantly. However, there is no guarantee that the DMP will encode and generalize the desired preferences.

5. Conclusions and Future Work

We presented a novel approach for learning and executing human preferences in robot object carrying tasks. Our user study showed fast convergence of the algorithm and a proof-of-concept study was detailed for generalizing path and velocity preferences. The efficiency and accuracy of our approach were validated in a real-world scenario. Our supplementary study compared the performance of our framework to two common methods from the literature, providing additional insights into the benefits and drawbacks caused by the structural differences between the methods. Both in the user study and in the supplementary study, a single informative feedback sufficed (in all cases except one) to capture the human preferences. In the user study, this was tested without prescribing a preference to the users. Our framework was in most cases successful in generalizing these preferences to previously unseen scenarios. Our results support that our model contributes to personalized planning of object carrying tasks with low interaction effort.

Future user studies comparing our method (with just path preferences) to PHI [4,6] could lead to useful insights into people's preferences on iterative versus online learning. Further research could consider a combination of our method and PHI that would benefit from the advantages of both, namely achieving generalization both in-task and over new task instances through learning from online interactions. Following this, the trajectory model we used to make the problem tractable is quite simplistic and does not describe human motion behavior very well. Future research can aim to replace this model with a library of motion primitives generated from demonstrations to better capture the shape of the trajectories. More accurate trajectory models can enable the extension of the framework to settings where the human and robot come into contact with each other through a shared object (physical human–robot collaboration). Furthermore, whether more complex non-linear formulations of the reward function using Gaussian Processes [32] or Neural Networks [16], and/or learning them from user input [29,30], can effectively capture context-aware preferences without the need for rigorous feature engineering should be

studied. We believe the presented framework is especially effective in collaborative settings, where knowledge of the preferences of a partner is essential in the execution of the task.

Author Contributions: A.A., L.v.d.S., L.P. and J.K. developed the concept and methods. The programming was mainly performed by A.A., while L.v.d.S. modified PHI for the supplementary comparison study. A.A. and L.v.d.S. contributed to the experiments and data analysis and wrote the first draft of the paper. L.v.d.S., L.P. and J.K. revised the paper. All authors read and approved the submitted version. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the European Research Council Starting Grant TERI "Teaching Robots Interactively" (project reference 804907) and the European Space Agency through the project "Rhizome: Off-Earth Manufacturing and Construction". This study received funding from the Honda Research Institute Europe.

Institutional Review Board Statement: The study was conducted in accordance with the Declaration of Helsinki, and approved by the Human Research Ethics Committee at Delft University of Technology on 6 September 2021.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The datasets generated for the performed user and supplementary study are available on request. Please contact the corresponding author.

Acknowledgments: The authors thank Andreea Bobu for her feedback on the paper.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

DoF	Degree of Freedom
LfD	Learning from Demonstration
DMP	Dynamic Movement Primitive
IRL	Inverse Reinforcement Learning
POMDP	Partially Observable Markov Decision Process
NASA	National Aeronautics and Space Administration
TLX	Task Load Index
PHI	Physical Human Interaction as in Bajcsy et al. [4]
MDPI	Multidisciplinary Digital Publishing Institute

References

1. Jain, A.; Sharma, S.; Joachims, T.; Saxena, A. Learning preferences for manipulation tasks from online coactive feedback. *Int. J. Robot. Res.* **2015**, *34*, 1296–1313. [CrossRef]
2. Duchaine, V.; Gosselin, C.M. General model of human–robot cooperation using a novel velocity based variable impedance control. In Proceedings of the Second Joint EuroHaptics Conference and Symp. on Haptic Interfaces for Virtual Environment and Teleoperator Systems , Tsukuba, Japan, 22–24 March 2007 .
3. Peternel, L.; Petrič, T.; Oztop, E.; Babič, J. Teaching robots to cooperate with humans in dynamic manipulation tasks based on multi-modal human-in-the-loop approach. *Auton. Robot.* **2014**, *36*, 123–136. [CrossRef]
4. Bajcsy, A.; Losey, D.P.; O'Malley, M.K.; Dragan, A.D. Learning robot objectives from physical human interaction. In Proceedings of the Conference on Robot Learning, Mountain View, CA, USA, 13–15 November 2017.
5. Losey, D.P.; O'Malley, M.K. Learning the correct robot trajectory in real-time from physical human interactions. *ACM Trans. Hum.-Robot Interact.* **2019**, *9*, 1–19. [CrossRef]
6. Losey, D.P.; Bajcsy, A.; O'Malley, M.K.; Dragan, A.D. Physical interaction as communication: Learning robot objectives online from human corrections. *Int. J. Robot. Res.* **2022**, *41*, 20–44. [CrossRef]
7. Ijspeert, A.J.; Nakanishi, J.; Schaal, S. Movement imitation with nonlinear dynamical systems in humanoid robots. In Proceedings of the IEEE International Conference on Robotics and Automation, Washington, DC, USA, 11–15 May 2002.
8. Kulvicius, T.; Biehl, M.; Aein, M.J.; Tamosiunaite, M.; Wörgötter, F. Interaction learning for dynamic movement primitives used in cooperative robotic tasks. *Robot. Auton. Syst.* **2013**, *61*, 1450–1459. [CrossRef]

9. Gams, A.; Petrič, T.; Do, M.; Nemec, B.; Morimoto, J.; Asfour, T.; Ude, A. Adaptation and coaching of periodic motion primitives through physical and visual interaction. *Robot. Auton. Syst.* **2016**, *75*, 340–351. [CrossRef]
10. Nemec, B.; Likar, N.; Gams, A.; Ude, A. Human robot cooperation with compliance adaptation along the motion trajectory. *Auton. Robot.* **2018**, *42*, 1023–1035. [CrossRef]
11. Ewerton, M.; Maeda, G.; Kollegger, G.; Wiemeyer, J.; Peters, J. Incremental imitation learning of context-dependent motor skills. In Proceedings of the 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), Cancun, Mexico, 15–17 November 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 351–358.
12. Wirth, C.; Akrour, R.; Neumann, G.; Fürnkranz, J. A survey of preference-based reinforcement learning methods. *J. Mach. Learn. Res.* **2017**, *18*, 1–46.
13. Ratliff, N.D.; Bagnell, J.A.; Zinkevich, M.A. Maximum margin planning. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006.
14. Ziebart, B.D.; Maas, A.L.; Bagnell, J.A.; Dey, A.K. Maximum entropy inverse reinforcement learning. In Proceedings of the AAAI, Chicago, IL, USA, 13–17 July 2008.
15. Jeon, H.J.; Milli, S.; Dragan, A. Reward-rational (implicit) choice: A unifying formalism for reward learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 4415–4426.
16. Ibarz, B.; Leike, J.; Pohlen, T.; Irving, G.; Legg, S.; Amodei, D. Reward Learning from Human Preferences and Demonstrations in Atari. *arXiv* **2018**, arXiv:1811.06521.
17. Bıyık, E.; Losey, D.P.; Palan, M.; Landolfi, N.C.; Shevchuk, G.; Sadigh, D. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *Int. J. Robot. Res.* **2022**, *41*, 45–67. [CrossRef]
18. Kirby, R.; Simmons, R.; Forlizzi, J. Companion: A constraint-optimizing method for person-acceptable navigation. In Proceedings of the 18th IEEE International Symp. on Robot and Human Interactive Communication, Toyama, Japan, 27 September–2 October 2009.
19. Kretzschmar, H.; Spies, M.; Sprunk, C.; Burgard, W. Socially compliant mobile robot navigation via inverse reinforcement learning. *Int. J. Robot. Res.* **2016**, *35*, 1289–1307. [CrossRef]
20. Shivaswamy, P.; Joachims, T. Coactive learning. *J. Artif. Intell. Res.* **2015**, *53*, 1–40. [CrossRef]
21. Palan, M.; Landolfi, N.C.; Shevchuk, G.; Sadigh, D. Learning reward functions by integrating human demonstrations and preferences. *arXiv* **2019**, arXiv:1906.08928.
22. Fahad, M.; Chen, Z.; Guo, Y. Learning how pedestrians navigate: A deep inverse reinforcement learning approach. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain, 1–5 October 2018.
23. Zucker, M.; Ratliff, N.; Dragan, A.D.; Pivtoraiko, M.; Klingensmith, M.; Dellin, C.M.; Bagnell, J.A.; Srinivasa, S.S. CHOMP: Covariant Hamiltonian Optimization for Motion Planning. *Int. J. Robot. Res.* **2013**, *32*, 1164–1193. [CrossRef]
24. Vasquez, D.; Okal, B.; Arras, K.O. Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014.
25. MathWorks. Waypoint Trajectory Generator, 2018. Available online: https://www.mathworks.com/help/fusion/ref/waypointtrajectory-system-object.html (accessed on 3 August 2021).
26. Hart, S.G.; Staveland, L.E. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in Psychology*; Elsevier: Amsterdam, The Netherlands, 1988; Volume 52, pp. 139–183.
27. Akgun, B.; Cakmak, M.; Jiang, K.; Thomaz, A.L. Keyframe-based learning from demonstration. *Int. J. Soc. Robot.* **2012**, *4*, 343–355. [CrossRef]
28. Mülling, K.; Kober, J.; Kroemer, O.; Peters, J. Learning to select and generalize striking movements in robot table tennis. *Int. J. Robot. Res.* **2013**, *32*, 263–279. [CrossRef]
29. Bobu, A.; Wiggert, M.; Tomlin, C.; Dragan, A.D. Inducing structure in reward learning by learning features. *Int. J. Robot. Res.* **2022**, *41*, 497–518 . [CrossRef]
30. Katz, S.M.; Maleki, A.; Bıyık, E.; Kochenderfer, M.J. Preference-based learning of reward function features. *arXiv* **2021**, arXiv:2103.02727.
31. Calinon, S.; Lee, D. Learning Control. In *Humanoid Robotics: A Reference*; Vadakkepat, P., Goswami, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2019; pp. 1–52. [CrossRef]
32. Bıyık, E.; Huynh, N.; Kochenderfer, M.J.; Sadigh, D. Active Preference-based Gaussian Process Regression for Reward Learning. *arXiv* **2020**, arXiv:2005.02575.