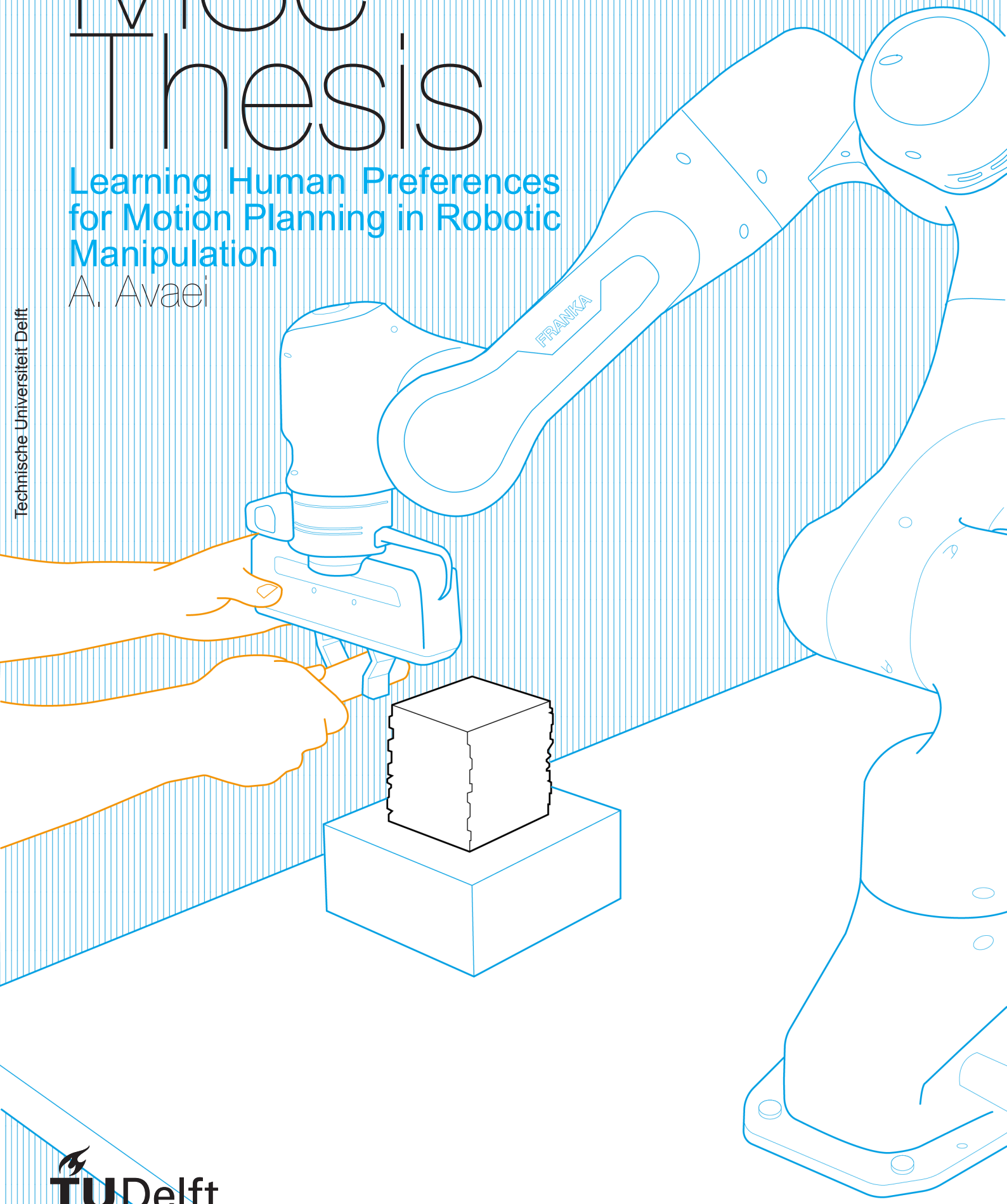
MSc Thesis

Learning Human Preferences for Motion Planning in Robotic Manipulation

A. Avaei

Technische Universiteit Delft



TU Delft

Learning Human Preferences for Motion Planning in Robotic Manipulation

by

A. Avaei

to obtain the degree of

Master of Science
in Mechanical Engineering at the Delft University of Technology,
to be defended publicly on Thursday November 25, 2021 at 10:00 AM.

Student number: 5048427
Supervisors: Dr. ing. J. Kober
Dr. L. Peternel
Ir. L.F. van der Spaa

An electronic version of this thesis is available at: http://repository.tudelft.nl/.

TU Delft Delft University of Technology

Preface

Conducting my master's thesis at the Department of Cognitive Robotics for the past 8 months has been one of the most fulfilling experiences of my life.

I would like to extend my sincere gratitude towards my supervisors Jens Kober and Luka Peternel for their enthusiastic guidance and constructive advice throughout the thesis. I am also extremely grateful to have had Linda van der Spaa as a mentor, showing me the ropes for how to conduct research. Our engaging discussions were instrumental to the development of the project, and I deeply appreciate her belief in my work. I also wish to thank Giovanni Franzese for getting me started with the robot controllers and providing me with insightful suggestions.

I had great pleasure of working alongside my friends Francesco, Alvaro, Anna, Irene and Sven at the Cognitive Robotics Lab. Many thanks to them for the the occasional brainstorms and making the lab a pleasant environment to be in.

Completion of this thesis would not have been possible without the support of my second family in the Netherlands, Francesca, Joseph, Katherina and Stella, who were there for me during the highs and lows. Finally, a heartfelt thank you to my dear family back home for their relentless support throughout the years.

A. Avaei
Delft, November 2021

Contents

1

Paper

Learning Human Preferences for Motion Planning in Robotic Manipulation

Armin Avaei
Supervisors: Linda van der Spaa, Luka Peternel, and Jens Kober

***Abstract*—Humans often demonstrate diverse behaviours 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 such preferences into planning of trajectories for robotic manipulators. We first learn reward functions that represent the users path and motion preferences from kinesthetic demonstration. We then use a discrete-time trajectory optimization scheme to produce trajectories that adhere to both task requirements and user preferences. Our work goes beyond the state of art by achieving generalization of preferences to new task instances, and designing a large feature set that enables capturing of the dynamical aspects of the manipulation, such as preferences about the timing of motion. We implement our algorithm on a Franka Emika Panda 7-DoF robotic arm, and present the functionality and flexibility of our approach by testing it 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.**

***Index Terms*—Learning from demonstration, planning with preferences, human-robot cooperation**

I. Introduction

AUTONOMY is increasingly being discussed under the aspect of cooperation. A gentler breed of robots, "cobots", have started to appear in factories and workshops, working together with humans. One challenge in deployment of such robots is producing desirable trajectories for manipulation tasks. A desirable trajectory not only meets the task constraints (i.e. collision-free movement from start to goal), but it also adheres to human user's preferences. Such preferences may vary between users, environments and tasks. Therefore, 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 [2].

We address the challenge of learning human preferences in an individual context when the robot plan does 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). In such cases the user prefers the robot to execute the task in a different manner than originally planned. Fig. 1 illustrates an example where the user demonstrates multiple preferences by demonstrating a preferred trajectory.

Authors are with the Department of Cognitive Robotics, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands (email: s.avaei@student.tudelft.nl, {L.F.vanderSpaa, J.Kober, L.Peternel}@tudelft.nl).

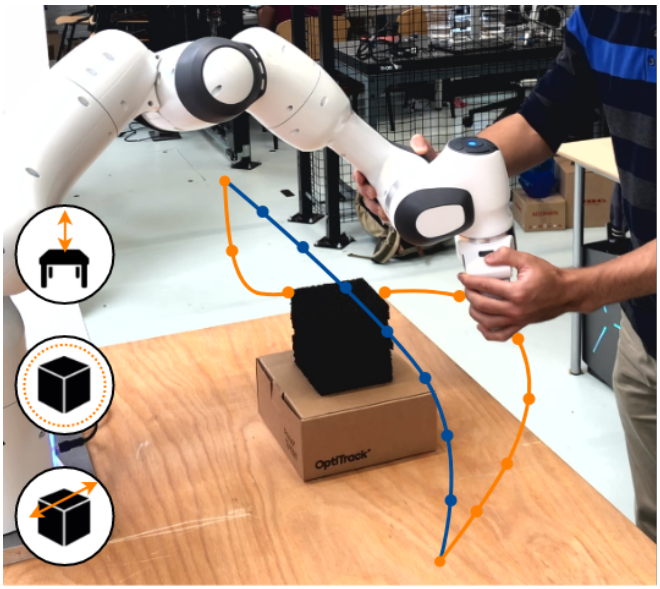


Fig. 1. We show how to leverage demonstrations as means of understanding the human's preferences in a manipulation task. The robot originally plans the blue trajectory without knowledge of human preferences. The user demonstrates the orange trajectory which in this instance contains the following preferences: "Stay close to the table surface", "Keep larger distance from the obstacle", and "Pass on the far side of the obstacle". We develop a method for learning and generalizing these preferences to new task instances.

One method for adhering to such preferences is to achieve adaptability on the lower control layer. To this extent, several control-based approaches have been investigated since the early 1990s [3]–[5]. These approaches share the same variable impedance control structure that enables the robot to render an apparent impedance when a user applies a force in their desired direction of motion. While such strategies can ensure safe and responsive adaptation, they suffer from being purely reactive (i.e. they do not remember the 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 improvement of the corrected trajectory [6]. Hence, there is a need for robots with knowledge of human's desired trajectories. In our approach, this knowledge is encoded as a set of parameters that are incrementally updated based on the corrected trajectory.

In this regard, Learning from Demonstration (LfD) is an active area of research which enables robots to encode human demonstrated trajectories [2], [7], [8]. LfD frameworks have the advantage of enabling non-robotic experts to naturally teach trajectories to robots. A widespread approach in LfD

is Dynamic Movement Primitives [9]. In addition to encoding trajectories, DMPs have the ability to adapt the learned path via updating an interactive term in the model [10], [11]. Additionally, they can adapt the velocity of the motion by estimating the frequency and the phase of a periodic task [12], or learning a speed scaling factor [13]. As a result, they are able to implicitly capture human path and velocity preferences. However, such parameters and factors still do not contain any knowledge about the context of the task and why the trajectory was adjusted in the first place. Hence, such adaptations are limited to one specific task and user, and trajectory corrections are often necessary. Accordingly, these methods fail to generalize user preferences to new scenarios due to the robot's lack of a higher-level understanding of the human actions. In comparison, our approach pairs parameters with features that capture contextual information (e.g. distance to obstacle), and utilises this information to find an optimal solution in new scenarios.

One way of achieving such generalization is to learn a model of what makes a trajectory desirable for a person. An approach to this problem is Inverse Reinforcement Learning (IRL), which is a paradigm for learning reward functions from expert demonstrations [14]. Recently, to alleviate the issue of dependence on expert knowledge, preference-based learning algorithms have been proposed. These algorithms learn reward functions by querying users for pairwise comparisons, asking them to express preferences between different trajectories [15]. Work in this direction has explored a model-free approach to learning complex non-linear reward functions in the Atari domain [16], but such an approach suffers from requiring many queries to learn from, which is time-intensive. Therefore, we focus on approaches with a simple linear reward structure. Coactive learning [17] is a method to learn such a reward function. Unlike traditional IRL methods [18], [19], coactive learning does not rely on optimal demonstrations (i.e. has an upper boundary on regret, leaving room for noisy and imperfect user feedback). It also benefits from being an online learning algorithm (i.e. the system can learn incrementally from sequential feedback).

An adapted version of coactive learning was applied in [1] to learn preferences over trajectories in object manipulation tasks. To this extent, users iteratively ranked trajectories proposed by the system. Trajectories were sampled using the Rapidly-exploring Random Tree (RRT) algorithm and the highest scoring trajectory based on the learned reward was selected to be the top trajectory. Using randomized sampling of trajectories limits the generalization capability and quality of preference quarries, which in turn increases number of iterations of feedback necessary for convergence. In contrast, we focus on learning from a few informative feedback, and give special attention to the issue of trajectory sampling by employing a model-based trajectory optimization. This facilitates the extraction of maximum amount of information out of each feedback and generalization of preferences to entirely new contexts. Furthermore, our approach enables learning of preferences related to velocity.

A similar learning algorithm was used in [6] to adapt the robot's trajectory to user's preferences based on force feedback. While this approach employed a trajectory optimization scheme, it only achieved in-task generalization (i.e. preferences were only applied to to future time steps of the current task). We go further by achieving generalization to new task instances, while still formulating the problem in the form of trajectory optimization.

In this paper, we propose a novel framework for optimizing trajectories in manipulation tasks that meet the user's path and velocity preferences. The objective function for the optimization partly comprises a human preferences reward function, and a fixed robot objective function that ensures the safety and efficiency of the trajectories. The approach takes a full demonstrated trajectory as the feedback for the learning model, comparing it against the robot's previous plan at each step. A minimum acceleration trajectory model is used to significantly reduce the size of the task space in which we plan trajectories in, hence increasing the optimization efficiency. Finally, we design an extensive set of features that correspond to a variety path and velocity preferences. We evaluate the proposed method both in simulation and user studies on a Franka Emika Panda 7-DoF robotic arm, and we show that the robot can learn and generalise human preferences from as few as a single demonstration.

In summary, this paper contributes by introducing a methodology that is able to: 1) learn different human preferences from a single informative feedback; 2) optimally estimate the preferred trajectory in new task instances by exploiting a trajectory model; 3) execute the learned behaviour in a safe and compliant manner. For this, we combined existing IRL, optimisation, and control methods in a novel way.

The rest of the paper is organized as follows: in Sec. II, we discuss the algorithm and methodology in detail. Sec. III and IV show our experiments and results in simulations and a user study, and Sec. V provides a comprehensive discussion on the results. Finally, we present the concluding remarks and future work in Sec. VI.

II. Method

The problem can be defined in the following manner. Given a context $\mathcal{C}$ which describes the start, goal, and obstacle positions, the robot has to determine the trajectory $\boldsymbol{\xi} = \boldsymbol{s}^{1:T} \in \Xi$ (i.e. a 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.

We adopt the formalisation of this problem from [6] as a Partially Observable Markov Decision Process (POMDP), where the true reward functions are known by the user and not the robot. This is on the account that our reward functions have parameters that are part of the hidden state (i.e. not directly observable), and the trajectories provided by the user are observations about these parameters. Solving POMDPs is challenging in robotics, where the control space is very complex and high-dimensional. Therefore, we simplify the problem through approximation of the policy by separating the planning and control, and treating it as an optimization problem. Furthermore, we make the problem tractable by

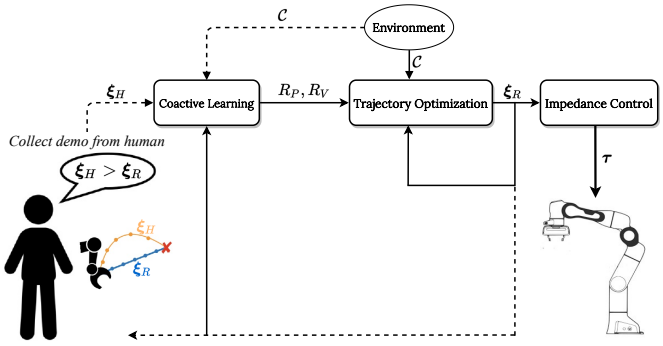


Fig. 2. Schematic overview of the framework. The human user provides demonstrations, which are used to learn a distribution over reward functions via coactive learning. Then, we use the learned rewards to adapt the trajectory planning strategy of the robot and optimize trajectories to maximise the human preferences rewards. We then execute the resulting trajectory using an impedance controller. We repeat this process of querying the human for preferred trajectories until convergence, at which point the human can be taken out of the loop.

going from the state space of the manipulator to a space of viable smooth trajectories.

The resulting framework, as depicted in Fig. 2, first learns the appropriate reward functions, then plans a trajectory that maximises 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 planning and velocity planning in the learning and optimization steps. As a result, users have the flexibility to demonstrate their path and velocity preferences either simultaneously or in separate demonstrations.

A. Learning human reward functions from demonstration

We follow previous IRL work [1], [18], [20] in assuming that the reward functions are a linear combination of features ϕ. Accordingly, we define the path and velocity reward functions R_P and R_V as

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

$$R_V(\dot{x}(r);\mathcal{C},\boldsymbol{\theta}_{HV}) = \boldsymbol{\theta}_{HV}^T \cdot \boldsymbol{\Phi}_V(\dot{x}(r);\mathcal{C}), \tag{1b}$$

where $\boldsymbol{\theta}_{HP}$ and $\boldsymbol{\theta}_{HV}$ denote the unknown weights that respectively encapsulate the human path and velocity preferences. In case of the velocity reward, we divide the trajectory into segments (i.e. range of samples) with equal lengths indicated by r, with $\dot{x}(r)$ indicating the average velocity in each segment. $\boldsymbol{\Phi}_P$ and $\boldsymbol{\Phi}_V$ represent the total path and velocity feature counts along the trajectory, such that:

$$\boldsymbol{\Phi}_P = \sum_{k=1}^{N} \boldsymbol{\phi}_P\left(\boldsymbol{x}(k);\mathcal{C}\right), \tag{2a}$$

$$\boldsymbol{\Phi}_V = \sum_{r=1}^{M} \boldsymbol{\phi}_V\left(\dot{x}(r);\mathcal{C}\right), \tag{2b}$$

where N is the fixed number of samples every path is resampled to, and M is the number of segments the trajectory is divided to for forming the velocity reward. Importantly, to have comparable rewards all trajectories need to be consisted of the same number of states, hence the resampling to N. The velocity however, inherently affects the number of samples within a trajectory, which is why we opt for dividing 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 briefly describe them in the next subsection and in more detail in Appendix A. Note that the uncertainty over $\boldsymbol{\theta}_{HP}$ and $\boldsymbol{\theta}_{HV}$ is what makes the problem a POMDP. The trajectory demonstrated by the human is taken as an observation about these unknown parameters [6].

Assuming that the human behaviour is approximately optimal with respect to the true reward, we use a variant of coactive learning introduced in [6] to learn the weights in $\boldsymbol{\theta}_{HP}$ and $\boldsymbol{\theta}_{HV}$. However, instead of updating the weights based on an estimate of human's intended trajectory from physical interactions, we use a full trajectory from kinesthetic demonstration performed by the human after each task execution. This provides the algorithm with high-quality feedback as training data which leads to fast convergence of weights as shown in III-A. The update rule is then given by

$$\boldsymbol{\theta}_H^{i+1} = \boldsymbol{\theta}_H^i + \alpha\left(\boldsymbol{\Phi}\left(\boldsymbol{\xi}_H^i;\mathcal{C}^i\right) - \boldsymbol{\Phi}\left(\boldsymbol{\xi}_R^i;\mathcal{C}^i\right)\right), \tag{3}$$

where $\boldsymbol{\theta}_H$ can either correspond to weights for the path or velocity preferences (we update these parameters separately), i denotes the iteration number, $\boldsymbol{\xi}_H^i$ and $\boldsymbol{\xi}_R^i$ are respectively the human demonstrated trajectory and the robot's current optimized trajectory, and $\alpha \in (0, 1]$ is the learning rate. Note that in case of updating the path preferences, we only use the position part of the state, and in case of updating velocity preferences we only use the velocities. Intuitively, the update rule is an online gradient that shifts the weights in the direction of the human's intended feature count. As we use full new demonstrations at each iteration $\boldsymbol{\xi}_H^i$, this update rule is analogous to an online version of the Maximum Margin Planning algorithm [18], [6].

B. Features and Costs

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, whereas the robot objectives consist of costs that define a basic behaviour for the robot in absence of any human rewards. Moreover, robot objectives counter-balance the effect of human rewards in the optimization. While we learn the weights in the human rewards, the weights in the robot objectives are hand-tuned. In this section we first describe the features associated with the human rewards, and then we describe the robot objectives.

The human preferences are captured via the four features listed below (see Fig. 1 for an example of the listed path preferences). We chose these features as they characterize dominant behaviours in manipulation applications that could vary from one user to another based on their preferences. Additionally, the features selected act along different dimensions of the

workspace, creating a complete definition for the behaviour of trajectories.

Height from the Table: To capture the preference related to the height from the table on a range of 'low' to 'high', we designed a sigmoid function centered at a 'medium' height above the table.

Distance to the Obstacle: We encode the user's preferred distance to the obstacle on a range of 'close' to 'far' using an exponential feature. The value of this feature decreases as the radial distance of the end-effector to the obstacle increases. Beyond a predefined threshold the value of this feature drops to 0 and it no longer influences the behaviour of the optimization.

Obstacle Side: To capture which side of the obstacle to pass from we created a tangent hyperbolic function centered at 0 lateral distance from the obstacle. We define this feature on a range of 'close side' (the side of the obstacle closer to the robot) to 'far side' (the side of the obstacle far from the robot).

Velocity: To encapsulate the user's velocity preferences, we adopt a different approach using a discretized linear combination of Radial Basis Functions (RBFs). For each segment r, we map the average velocity onto an even distribution of RBFs over a range of velocities ($\dot{x}_{min}$ - $\dot{x}_{max}$). The RBFs are given by:

$$\psi_j(\dot{x}(r)) = e^{-(\varepsilon \dot{x}(r) - c_j)^2}, \tag{4}$$

where ε is the shape variable that defines the width, and c_j defines the center of the j^{th} RBF, with $j = 1, 2, \cdots, n$ (we selected n=9).

Inspired by [21], we discretize the above feature to two bins, based on the distance of the center of each segment to the obstacle d. Hence, we have the two cumulative feature vectors: $\mathbf{\Phi}_{V1}$ for $d \in [0, d_r)$ and $\mathbf{\Phi}_{V2}$ for $d \in [d_r, \infty)$. This allows us to approximate the speed of motion separately in areas considered to be 'close' to the obstacle and areas considered to be 'far' from the obstacle. However, an issue might arise that the two trajectories might not have the same number of segments in each bin (e.g. $\boldsymbol{\xi}_R$ having two segments close to the obstacle while $\boldsymbol{\xi}_H$ only having one segment within the d_r threshold). In such a case, we employ feature imputation using the mean of the available values.

The robot's objectives are composed of the following costs and rewards:

Path Efficiency Cost: This cost is calculated as the total length of a trajectory and is essential in counter balancing 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.

Collision Avoidance Cost: This cost is defined based on the obstacle cost formulation in [22], and increases exponentially once the distance to the obstacle drops below a predefined threshold.

Robot Velocity: We opt for a robot velocity reward that achieves a low and safe velocity in absence of human velocity preferences. This reward is defined based on (4) and is centered at a low velocity. Furthermore, in IRL it is beneficial to learn how people balance other features against a default reward [23].

In the following subsection we will describe how the learned rewards can be combined with robot objectives to produce human-centered trajectories.

C. Motion planning via trajectory optimization

We discuss the problem of motion planning in two parts. First, we address the optimization of the path which solely defines the position of the trajectory samples 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 entire Cartesian task-space would be very complex and inefficient. As a result, we employ a trajectory planning algorithm [24] that interpolates between a set of waypoints with piecewise clothoid curves to create a trajectory. This algorithm minimizes the acceleration which results in a smooth and realistic motion. Hence, 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 the corresponding time vector $\mathbf{t}$:

$$\boldsymbol{\xi} = f(\mathbf{p}, \mathbf{t}_P). \tag{5}$$

We consider three waypoints $\mathbf{p} = [\mathbf{p}^s \ \mathbf{p}^m \ \mathbf{p}^g]^T$, corresponding respectively to the start position, an arbitrary position within the path, and the goal position. We further simplify the problem by fixing the time vector to $\mathbf{t}_P = [0 \ \frac{D(\mathbf{p}^m)}{D(\mathbf{p}^g)} \ t^g]^T$, where $D(\mathbf{p})$ indicates the Euclidean distance of a waypoint to the start position, and t^g is a fixed duration of time that, at this step, we assume all trajectories take to finish[1]. An uneven distribution of waypoints would lead to bias in the reward value. Setting up the time vector in this manner ensures a constant velocity is achieved throughout the trajectory, which results in an even distribution of interpolated samples within the path. Furthermore, in this manner trajectories can be sampled only as a function of waypoint positions $\boldsymbol{\xi} = f(\mathbf{p})$.

The path optimization problem can then be solved by finding the optimal waypoint vector $\mathbf{p}^*$ using the following non-linear program formulation:

$$\begin{aligned} &\mathbf{p}^* = \arg\min_{\mathbf{p}} \Big(R_P(\mathbf{p}; \mathcal{C}, \boldsymbol{\theta}_{HP}) + \boldsymbol{\theta}_{RP}^T \cdot \mathbf{\Phi}_{RP}(\mathbf{p}; \mathcal{C}) \Big), \\ &\text{subject to:} \\ &\boldsymbol{h}(\mathbf{p}) = \mathbf{0}, \\ &\mathbf{p}_{\text{low}} \leq \mathbf{p} \leq \mathbf{p}_{\text{upp}}. \end{aligned} \tag{6}$$

Here, the objective function is composed of the path reward function and the robot's path objective which is a linear combination of predetermined weights $\boldsymbol{\theta}_{RP}$ and the aforementioned path cost functions $\mathbf{\Phi}_{RP}$. The equality constraint is in place to ensure the start and goal positions are met. As a result, we are effectively searching for the waypoint $\mathbf{p}^m$ that maximises 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.

[1]Within the range of time duration we carry out manipulations for, the shape of the paths are not affected by the value of t^g, and therefore we assume the shape of the path to be independent of the velocity.

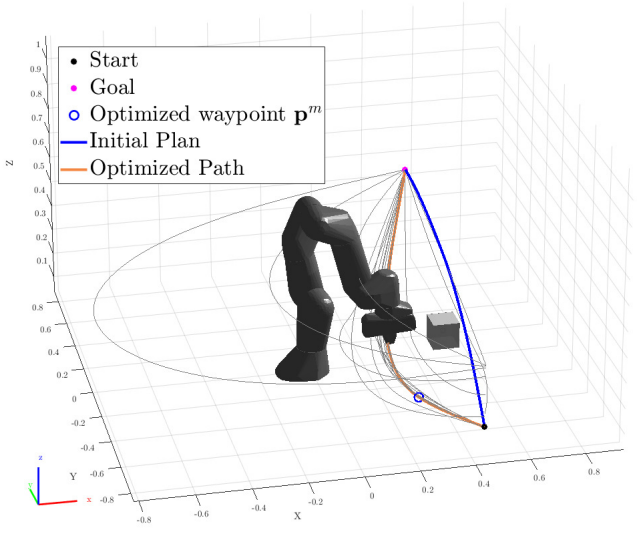


Fig. 3. An example of convergence towards the optimal path. The optimizer places $\mathbf{p}^m$ in different locations in the workspace to generate different paths. The paths explored by the optimizer are indicated in grey. The orange path indicates the output of the optimizer for path optimization,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

Once $\mathbf{p}^*$ is found, we can construct the full trajectory using $\boldsymbol{\xi}_P^* = f(\mathbf{p}^*, \mathbf{t}_P)$. Fig. 3 demonstrates an example of the convergence of the optimizer towards a path that adheres to the preferences shown in Fig. 1.

Having found $\boldsymbol{\xi}_P^*$ with the optimal path, we divide the trajectory into M segments similar to the learning step in Sec. II-A. Next, we sample M waypoints at the end of each segment. The positions of these waypoints are stored in $\mathbf{p}_V^* = [\mathbf{p}^1\ \mathbf{p}^2\ \cdots\ \mathbf{p}^M]^T$ and are fixed to maintain the shape of the trajectory. The corresponding timestamps for these waypoints are stored in $\mathbf{t} = [t^1\ t^2\ \cdots\ t^M]^T$, which are the variables we optimize for. Thus, trajectories sampled by the optimizer are only a function of the time vector $\boldsymbol{\xi} = f(\mathbf{t})$. By optimizing the time vector and keeping the segments fixed, we are essentially optimizing for the average velocity over each segment. The optimal time vector $\mathbf{t}^*$ can be computed from

$$\begin{aligned} &\mathbf{t}^* = \arg\min_{\mathbf{t}} \Big(R_V(\mathbf{t}; \mathcal{C}, \boldsymbol{\theta}_{HV}) + \theta_{RV} \cdot \phi_{RV}(\mathbf{t}; \mathcal{C}) \Big), \\ &\text{subject to:} \\ &\boldsymbol{g}(\mathbf{t}) \leq \mathbf{0}, \\ &\mathbf{t} \leq \mathbf{t}_{\text{upp}}, \end{aligned} \tag{7}$$

where the objective function is composed of the velocity preferences reward function R_V and the robot's velocity objective ϕ_{RV}, which provides a reward for manipulating objects at $\dot{x}_{robot}$ with a fixed weight θ_{RV}. The inequality constraint $\boldsymbol{g}(\mathbf{t})$ importantly acts as an upper and lower boundary on velocity, not allowing the time stamps to get too close or too far from each other (constraining the minimum and maximum speed over each segment to $\dot{x}_{min}$ and $\dot{x}_{max}$). The upper boundary on $\mathbf{t}$ acts as limit on the total duration of motion (limiting t^g to 30 seconds).

Finally, the trajectory that adheres to both the path and velocity preferences is constructed using:

$$\boldsymbol{\xi}_R = f(\mathbf{p}_V^*, \mathbf{t}^*). \tag{8}$$

D. Impedance control

We control the robot to track the desired trajectory at the end-effector via impedance control. This approach enables proactive position control, while simultaneously governing the compliance behaviour of the robot at the force level through $\mathbf{K}$ and $\mathbf{D}$, making it a safe strategy in cooperative environments. The impedance control law can be described as

$$\mathbf{f}_{\text{imp}} = \mathbf{K}\left(\mathbf{x}_r - \mathbf{x}\right) - \mathbf{D}\dot{\mathbf{x}}, \tag{9}$$

where $\mathbf{f}_{\text{imp}}$ is the impedance force, $\mathbf{K}$, $\mathbf{D} \in \mathbb{R}^{6\times6}$ are respectively the Cartesian stiffness and damping matrices, and $\mathbf{x}_r$ is the reference position outputted from (8). In our framework, the control damping matrix $\mathbf{D}$ is set to have a critically damped response $\mathbf{D} = 2\sqrt{\mathbf{K}}$ [25].

The total torque vector that is sent to the motor controller to achieve the desired impedance force and robot configuration can then be calculated from

$$\boldsymbol{\tau} = \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{q} + \boldsymbol{g}(\boldsymbol{q}) + \mathbf{J}^T\mathbf{f}_{\text{imp}}\ , \tag{10}$$

where $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are the joint angles and velocities, $\mathbf{J}$ is the robot's Jacobian matrix, $\boldsymbol{C}$ is the Coriolis and centrifugal vector, and $\boldsymbol{g}$ is the gravity vector.

During the kinesthetic demonstrations, the robot is set to compensate for gravity with $\mathbf{K} = \mathbf{0}$, which allows the users to easily manipulate the robotic arm. When a demonstration is intended to only update the speed behaviour of the robot, we set the robot compliant along a straight line and stiff in every other direction. This strategy is used in the user studies described in IV, and allows the human demonstrator to easily push or pull the robot's end-effector in a single direction with their desired speed.

Algorithm 1 Learning human preferences from kinesthetic demonstration

1: Record $\boldsymbol{\xi}_H^0 = \{\boldsymbol{x}(k), t_k\}_{k=1}^T$ using kinesthetic guiding and obtain context $\mathcal{C}^0$
2: Differentiate $\boldsymbol{x}(k)$ to obtain $\dot{\boldsymbol{x}}(k)$ and calculate average velocity of segments $\dot{x}(r)$
3: Initialize $\boldsymbol{\theta}_H^0$, $\boldsymbol{\theta}_R$ and $\boldsymbol{\xi}_R^0$
4: Set $i = 0$
5: **while** *cooperating* **do**
  if *Received Human Feedback* **then**
    $\boldsymbol{\theta}_H^{i+1} = \boldsymbol{\theta}_H^i + \alpha\left(\boldsymbol{\Phi}\left(\boldsymbol{\xi}_H^i; \mathcal{C}^i\right) - \boldsymbol{\Phi}\left(\boldsymbol{\xi}_R^i; \mathcal{C}^i\right)\right)$
  $\mathbf{p}^* \leftarrow$ Optimize$(\boldsymbol{\theta}_{HP}^{i+1}, \boldsymbol{\theta}_{RP}, \mathcal{C}^{i+1})$
  $\mathbf{t}^* \leftarrow$ Optimize$(\mathbf{p}^*, \boldsymbol{\theta}_{HV}^{i+1}, \theta_{RV}, \mathcal{C}^{i+1})$
  $\boldsymbol{\xi}_R = f(\mathbf{p}^*, \mathbf{t}^*)$
  $\boldsymbol{\tau} \leftarrow$ Impedance$(\boldsymbol{\xi}_R)$
  i=i+1

III. Simulation Study

A. Evaluation of Convergence

We test the convergence of our algorithm for 10 different simulated tasks in 5 different contexts (see Fig. B.1). To this extent, we study the convergence of the learning algorithm to a set of true weights $\boldsymbol{\theta}_{\text{true}}$, when an optimal demonstration is provided (i.e. the demonstration is the output of the trajectory optimization given the true weights). We use the expected cosine similarity alignment factor m [26] between the learned weights $\boldsymbol{\theta}_{HP}$ and the true weight distribution as a measure for convergence of the algorithm:

$$m = \mathbb{E}\left[\frac{\boldsymbol{\theta}_{HP} \cdot \boldsymbol{\theta}_{\text{true}}}{|\boldsymbol{\theta}_{HP}|\,|\boldsymbol{\theta}_{\text{true}}|}\right]. \tag{11}$$

$\boldsymbol{\theta}_{\text{true}}$ are hand-designed so that the trajectory output from the optimisation lies strictly within the boundaries in (6).

The results in Fig. 3 show that in most situations the algorithm converges remarkably fast. We further notice that when the weights do not converge, further iterations of feedback with the same trajectory can improve the results. However, for the case shown in light blue, we see a slight divergence with further updates. This is a particularly difficult case where the intended trajectory lies very close and above the obstacle (see Fig. B.1f). In such a region, the robot's collision avoidance objective heavily influences the result of the optimization, yet the learning algorithm does not take this information into account when updating $\boldsymbol{\theta}_{HP}$.

We further observe that for some of the cases the algorithm does not fully converge (c, h and j). Scenario c for instance, is associated with a situation in which the algorithm converges to an identical trajectory as the input, but does so with a slightly different distribution of weights (see Fig. B.1c). This is because there is redundancy in the weights, where in certain situations multiple weights push the trajectory in the same direction. For scenarios h and j the resulting trajectories are only slightly different in shape, but lie in the same region of workspace associated with the path preferences in the demonstration (see Fig B.1h and Fig. B.1j). While these results imply that our framework does not perform consistently in every situation in terms of convergence of weights, we believe the impact of such inconsistencies is minimal in terms of the user experience and satisfaction with the final trajectory produced. We further support this claim in Sec. IV.

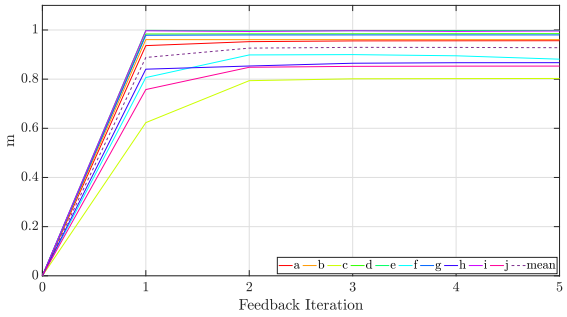


Fig. 4. The results of our convergence study, investigating how fast we can learn the distribution over $\boldsymbol{\theta}_{HP}$. The dashed line represents the mean. Refer to Fig. B.1 for the corresponding context and trajectories.

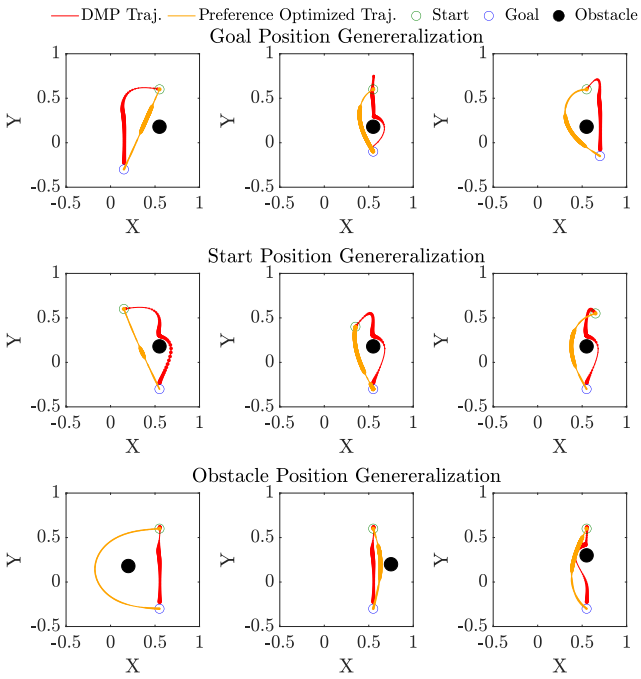


Fig. 5. We demonstrate generalisation by modifying the position of the goal, start and obstacle positions respectively from the top row to the bottom row. The red and orange trajectories respectively correspond to the output of the DMPs and our framework. The black, blue and green circles respectively represents the obstacle, goal and start positions. The thickness of the line indicates the inverse of normalised velocity (i.e. the thicker the line, the slower the trajectory).

B. Proof of Concept

To demonstrate the potential of our framework for reducing effort in cooperative settings, we compare it with the baseline of Dynamic Movement Primitives (DMPs). DMPs are a widespread LfD approach for encoding trajectories, capable of generalising to different initial/goal states (refer to Appendix B.2 for more details). We employ a variation of DMPs from [27], and modify it by adding a potential fields obstacle avoidance forcing term [28]. Hence, both methods have access to and consider the obstacle position for planning.

We consider a situation where the user has a preference for passing on the closer side of the obstacle due to existence of a wall on the other side that the robot is not aware of (similar to Fig. 3). Furthermore, we want to remain close to the obstacle, but to also slow down when we are in close proximity to it. We do a single kinesthetic demonstration containing all of these preferences, and use it as the input to our model. DMPs however do not require knowledge about the position of the obstacle in the learning phase. Therefore, in case of DMPs we provide a straight line demonstration from start to goal in absence of the obstacle. We still slow down in the middle area of the trajectory when demonstrating for the DMPs to provide both methods with similar velocity behaviours (see Fig. B.2b).

Fig. 5 displays trajectories reproduced by the two methods in 9 new situations. We can observe that the trajectory produced by our method (shown in orange) passes on the

'close' side of the obstacle in every case, where as the DMPs fail to realise this preference in 6 situations. Furthermore, the aforementioned velocity preference is more consistently achieved by our framework in every situation. This is because our model is aware that the path and velocity behaviours demonstrated are with respect to the the obstacle (context aware model), while DMPs only react to the obstacle in the trajectory reproduction phase. It should be noted that our framework does take up to two minutes of optimization (total for path and velocity), whereas the DMP trajectory is produced instantly. However, the human would have had to adapt the trajectory in all of the side preference failed cases, leading to higher effort.

This comparison illustrates how lack of a higher level knowledge about why the trajectory was demonstrated in a specific manner leads to failure in generalization to new task instances. These results emphasise the need for consideration of internal human models, such as our reward in (1), in LfD methods.

IV. User Study

To validate our framework we conduct two user experiments on a Franka Emika Panda 7-DoF robotic arm. Thereby we show a proof-of-concept of our approach in a real world robotic scenario with non-expert users[2]. In both experiments we use a set of three pick and place tasks in an agricultural setting. Each task has a distinct start, obstacle, and goal position. The task setups are shown in Fig. 6. The primary goal of each task was moving the tomatoes from the initial position to the goal without any collisions with the obstacle.

Each user first took approximately 10 minutes 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 different ranges of behaviours, in the first experiment we let the users freely define their path and velocity preferences. Once users were more familiar with the framework, in the second experiment we assessed how effectively could the users teach a set of predefined preferences to the robot. The overview of the whole study is provided in Fig. 7.

We recruited 14 participants (4 women and 10 men) between 23 and 36 years old (mean $= 26.8$, SD $= 3.6$), six of whom had prior experience with robotic manipulators, but none of whom had any prior exposure to our framework. We discuss each experiment in more detail in the following subsections.

A. User-Defined Preferences

Users demonstrate different trajectories depending on their experience and preferences. Hence, it is important to investigate how our framework performs when users can openly choose their set of preferences. We are 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). Furthermore, in terms of the user experience it is essential to investigate the acceptability and effort required from the user's perspective. Accordingly, our hypotheses for the first experiment are listed below:

H1. The proposed framework can generalize user preferences to new task instances.

H2. Users will believe the robot understood their preferences.

H3. Users feel a low level of interaction effort.

Procedure and Measures. Users first performed a demonstration in Task 1 for path preferences with the robot in gravity compensation mode. Notably, users could freely decide a wide range of behaviours with respect to each preference. 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 each side of the obstacle they would prefer to pass (i.e. we did not limit users to discrete preferences). The implication was that they could, for example, decide to pass right above the obstacle. This would then 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 above the obstacle in Task 1, and stiff in other directions. This line covered a range of distance to the obstacle in the range of "close" to "far". This allowed the users to focus on demonstrating their preferred speed without having to concern about the path taken. To reduce the time required for optimization, the method for learning and planning velocity preferences was modified from that described in Sec. II (see Appendix B.2).

Users were given maximum of 10 minutes per task and were instructed to provide corrections via additional kinesthetic demonstrations until they were satisfied with the result. However, users were informed that the speed of the trajectory can only be trained once via the initial demonstration and no further feedback could be given for velocity preferences. In order to evaluate the adaptability of the framework, once users were satisfied with the trajectory achieved in Task 3, they were asked to adapt their preferences to a different set of their choice and provide one additional demonstration.

Since we do not have direct access to the user's internal preferences, we quantify the quality of learning by asking each

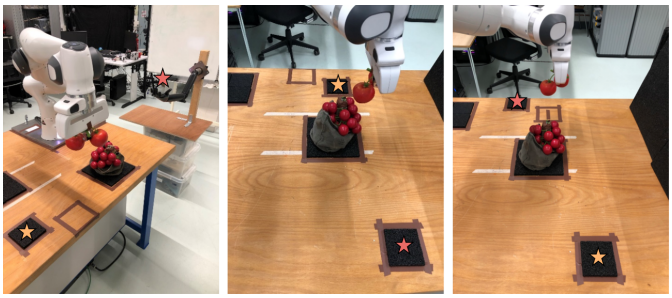


Fig. 6. From left to right tasks 1-3. The orange star indicates the start position, and the red start the goal position. The obstacle to be avoided is the bag of tomatoes. Task 1 and 2 shared the same starting positions, and Task 3 and 4 shared the same obstacle positions. Notice the difference in height of the goal position in Task 1 compared to Task 2 and Task 3.

[2]For video footage of the experiment, see: https://youtu.be/hhL5-Lpzj4M

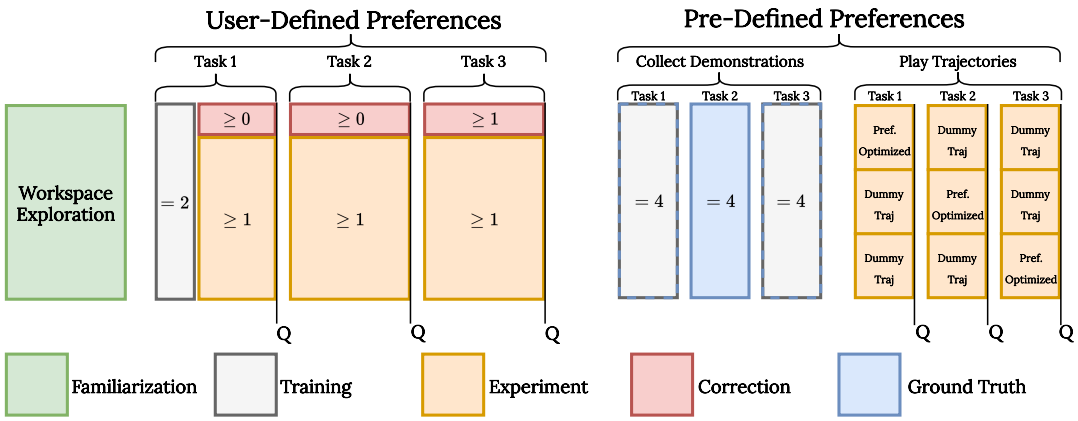
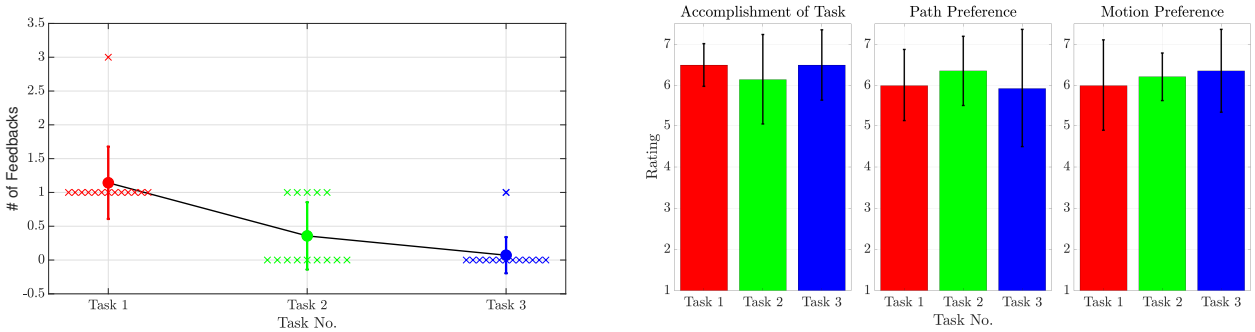


Fig. 7. The experimental protocol. Users started with workspace familiarization, then went through the first experiment assessing the performance of the framework in understanding their preferences. Finally, in the last experiment they provided ground truth demonstrations and evaluated the demonstrated trajectories in adhering to the set of pre-defined preferences. The order in which the dummy trajectories were shown to the users was different in every task. The 'Q' symbols indicate when participants were provided with questionnaires.



(a) (b)

Fig. 8. Results of the first experiment. (a) Average number of feedback provided to the system for each task. The dot represents mean score, the error bars represent standard deviation, and the crosses indicate individual data points. (b) Results of Likert questionnaire for the first resulting trajectory in every task (i.e. prior to any additional demonstrations) - the error bars correspond to standard deviation.

user to evaluate their own trajectories. After observing each trajectory, users filled out a subjective questionnaire to rate 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.
   If corrections were applied:
4) The robot learned from my correction.

We also evaluate the total number of times a user provided feedback to the system in each task as a measure of effort to achieve desired trajectories. Furthermore, at the end of this scenario they were asked to fill out the NASA Task Load Index (NASA-TLX) [29] for assessing workload. While we do not compare results with a baseline here, NASA-TLX is still appropriate since it can capture absolute results.

Results. While the majority of the users intuitively preferred to pass above the obstacle and to the middle, users still demonstrated a multitude of different "close/far" to obstacle, "left/right" side of obstacle, and "low/high" height from the table preferences. Similarly, for motion preferences, while the majority opted for a constant "medium" speed, both preference of going "slower when close to the obstacle" and "faster when close to the obstacle" were demonstrated at least once.

We present our results in Fig. 8. We observe that the average amount of feedback given to the system after the first task drops, with the majority of the users satisfied with the results of generalisation with just a single iteration of weight update (we count the training step in Task 1 as a feedback). This result is also reflected in Fig. 8b, showing that the users subjectively scored the first trajectory produced in every task consistently high, supporting the claim that the framework can generalise both path and motion preferences to new task instance. This provides strong evidence in favor of both **H1** and **H2**.

In regards to the workload associated with the method, the NASA-TLX results in Fig. 9 show that users experienced low mental and physical demand. Although kinesthetic teaching is normally associated with high effort, it is noteworthy that our frameworks effort scores remain on the lower side of the scale, with the exception of a few outliers. The user associated with 3 iterations of feedback in Task 1 (see Fig. 8a) corresponds to the high performance, frustration and effort scores. This user was particularly strict with their height preference, which the algorithm failed to capture, and over-corrected for in the updates. However, in general the results in Fig. 9 support **H3**.

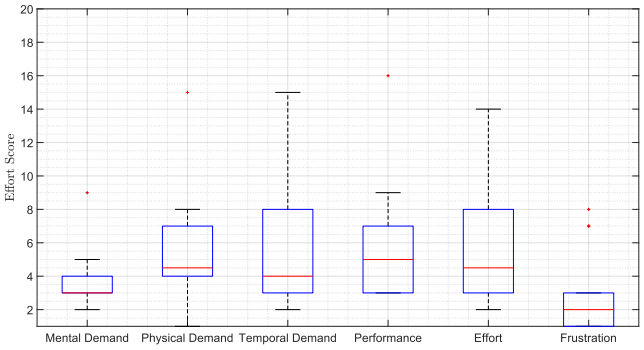


Fig. 9. Results of the NASA-TLX questionnaire after the first experiment.

B. Pre-Defined Preferences

To empirically evaluate the accuracy of trajectories planned by our framework, we conduct an experiment were we define the desired behaviour for the users. Correspondingly, all the users were asked to teach the robot 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.

As a measure for generalization, we evaluate if the method remains consistently accurate throughout the different tasks. Additionally, we investigate how well can users detect the instructed path preferences in trajectories. Therefore, we define the following hypotheses:

H4. The method remains consistently accurate throughout the 3 tasks.

H5. Users can clearly distinguish that the output of the framework is following the specified preferences.

Procedure and Measures. In this experiment, we first collected four demonstrations per task to obtain statistical measures on participants kinesthetic demonstration behaviour. For half of the participants, we trained the model on the mean of the four demonstrations from the first task, and for the other half we used the mean of data from the third task. This is to establish that we can still generalise between different tasks, even when changing the set used as the training data.

After collecting the data, users were shown 3 trajectories per task, one of which was the output of our framework, and the other two were pre-defined dummy trajectories (see Fig. 7). The dummy trajectories were designed to adhere to 2 out of 3 path preferences, and are shown in Fig. 10a. This allowed us to observe if users could distinguish our method's results compared to sub-optimal trajectories.

We take the mean of the trajectories demonstrated by the user to be the ground truth, with respect to which we can objectively measure the accuracy of our method. Hence, we compute the total Euclidean distance of samples within each trajectory to the samples within the mean of the four demonstrations in each task (we used N=80 in this experiment). Furthermore, we evaluate the total feature count along each trajectory and measure the error with respect to the ground truth in the feature space.

TABLE I. Average distance error of trajectory samples w.r.t the ground truth, normalized w.r.t distance of start to goal (in meters) - mean [min max].

	Task 1	Task 2	Task 3
Optim	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]

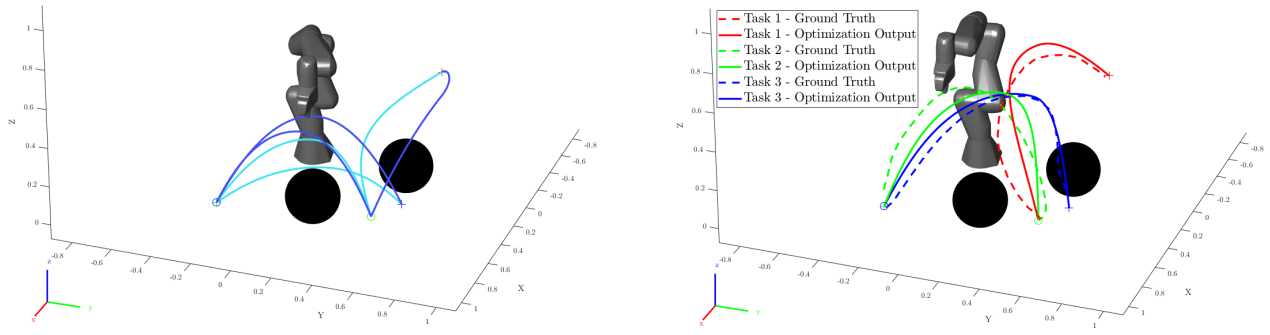
For our subjective measure, we asked users to rate each 7-point Likert scale based on the following statement:

- The robot adhered to the demonstrated preferences.

Results. Fig. 10b illustrates three successful trajectory generations by our method under the aforementioned path preferences for a single user. In this case, the model was trained from Task 3, and Task 1 and 2 represent the framework's generalisation. It is worth highlighting that in this experiment, both the human user and the robot are generalising the instructed preferences to new task instances. The difference is that the robot infers these preferences from a single trajectory input (mean of demonstrations), while the user uses the preference statements mentioned above. Therefore, as the users demonstrate slightly different trajectories based on their personal interpretation of high level statements such as 'high' and 'far', the robot attempts to capture and optimise for the personal amplitudes of each user for these preferences (i.e. one user's definition of 'high' is different from another).

Similar results were obtained from other users, and we show this by listing the min, max and mean of the average Eucledian distances between the samples in the resulting trajectories and ground truth trajectories in Table I. Note that these values have been normalised relative to the length of straight line trajectories in each task, in order to be able to compare trajectories in different tasks with various distances from start to goal. To provide an impression about the magnitude of these errors, the straight line trajectories in task 1-3 were respectively 1.08, 0.74 and 0.88 m. As expected, the trajectory produced by the optimisation has a notable lower distance error to the users demonstrations. Nevertheless, the results suggest that **H4** is only partially supported, as the mean distance error in Task 2 and 3 is slightly higher than in Task 1. Task 1 has the longest distance between the start and goal position, for which the framework seems to perform better.

The spider charts in Fig. 11 compare the mean error of the trajectories in feature space for all participants. It is evident that our optimisation results consistently occupy the least area on the charts in all three tasks, indicating that overall the optimised trajectories adhere to the preferences. However, the area corresponding to the optimised trajectories in Task 2 and 3 is slightly larger than in Task 1, showing the same trend of loss of performance in tasks with shorter lengths. Furthermore, we see that in Task 2 and 3, dummy trajectories occasionally perform slightly better for the 'obstacle side' and 'distance to the table' preferences. Nevertheless, in Fig. 12 we can see that users clearly score the output of our framework higher, which strongly supports **H5**. This indicates that users prefer all of the preferences to be approximately satisfied, rather than one of the preferences to be wrong while the others are accurate.



(a) (b)

Fig. 10. (a) The dummy trajectories designed for the second experiment. The blue trajectories do not correctly follow the 'obstacle side' preference, and the cyan trajectories do not follow the 'height from the table' preference. (b) Results of the second experiment for a single user. The dashed lines are the mean of human demonstrations taken as ground truth, and solid lines represent the robot trajectories, the black spheres the obstacles, and the circles and the plus signs are respectively the start and goal positions in each task. The framework uses the ground truth trajectory in blue to learn the human reward, but does not have access to the demonstrations in green and red. The robot successfully follows the pre-defined trajectories listed in Sec. IV-B.

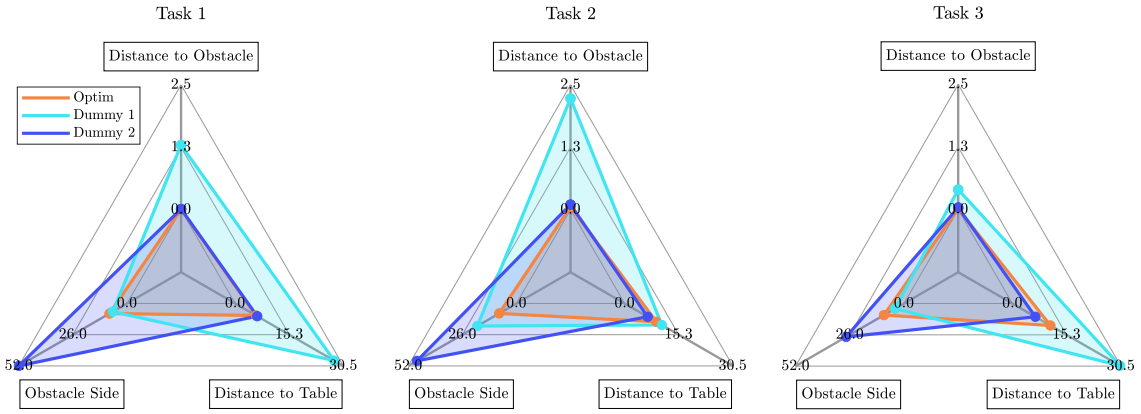


Fig. 11. Comparison of trajectories in terms of the error of total feature count of each path preference, with respect to the ground truth (i.e. smaller values for each axis are favoured). The orange chart corresponds to the output of our optimisation.

The best performing dummy trajectory was dummy 3-2 (solid blue trajectory for Task 3 in Fig. 10a), for which we can observe a correlation between a high rating in Fig. 12, low occupied area in Fig. 11, and relatively low distance error values in Table I. This correlation is also in support of **H5**, suggesting that non-expert users can intuitively recognize such preferences in trajectories.

V. Discussion

A. Mode of Demonstration

One of the advantages of the proposed method is that it learns fast. During the first part of the user study, participants spent on average 16.5 seconds of interacting with the robot before expressing satisfaction with the results. This is partially due to having access to (near)optimal kinesthetic demonstrations. This method of demonstration has been criticised as being challenging in applications involving high DoF manipulators [1], [30]. 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. Furthermore, as a result of separation of path planning and velocity planning, users are not required to provide a temporally consistent demonstration (i.e. velocity variations did not affect the result of path preference learning). These features made it significantly easier for the users to provide demonstrations, and this is reflected in the low mental and physical loads reported (see Fig. 9).

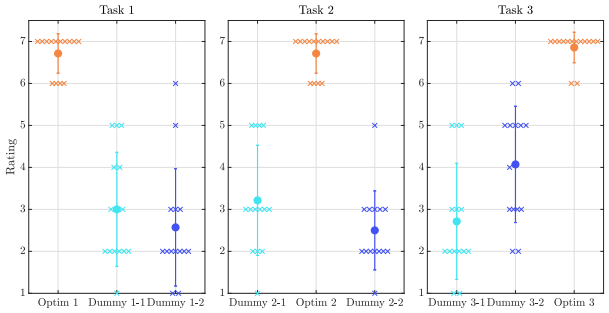


Fig. 12. Result of Likert questionnaire for experiment 2. Crosses indicate individual ratings, while the the dot and error bar respectively represent mean and standard deviation. Users clearly recognise and highly rate the output of the framework in terms of adhering to the path preferences.(Will be updated to include task colors)

B. Trajectory Optimization

The separation of path planning and velocity planning has additional benefits. Formulating the optimization as a multi-objective problem with both position and velocity features results in undesirable interaction of objectives. For instance, we observed that with such formulation, the optimizer attempts to modify the total length of the trajectory. This would then affect the distance between the samples and hence the velocity (e.g. when velocity features rewarded high speeds, the trajectory converged to a longer path). Conversely, path features with high rewards in specific regions of space were impacting the velocity. The optimizer slowed down the movement in such regions to increase the density of samples and consequently the overall reward. However, the separated format of trajectory optimization has the limitation that it can not account for dynamical quantities such as joint velocity and acceleration, and therefore efficiency of movements in robot's joint-space can not be considered.

A challenge with trajectory optimization with user defined objectives is that the outcome does not always align with task requirements. For instance, in case of a strong "stay close to the object" preference, a trajectory that is in collision briefly can have a lower average cost than a trajectory that never collides. Tuning of the collision weight can only partially solve this issue, as at a certain point this cost can interfere with the path preferences. In the first scenario of our experiments, we had one case where a collision happened. This was however fixed with one iteration of feedback, where the user slightly increasing the distance of the end-effector to the obstacle.

C. Accuracy Trade-off

The results of the first scenario of our user study in IV-A showed that users can intuitively use our method to quickly teach a wide range of preferences to the robot. In this context, the performance was defined based on satisfaction of users with the resulting trajectories, which depends on personal margins for error. While the results from the second scenario show that we do not always reproduce trajectories with the exact desired shapes in the workspace (see red trajectory in Fig. 10b), the results from the first scenario show that user's still deem these trajectories to be highly suitable in terms of task accomplishment and the preferences achieved (see Fig. 8b). State-of-art LfD methods are very capable of producing accurate and complex dynamic movements [31] [32]. However, we argue that in tasks such as manipulation where there are multiple ways of achieving the same goal, we can trade-off accuracy of motion for achieving more abstract planning propensities on a higher level (refer to [33] for a discussion on the need for extension of human-robot cooperation to the higher tactical layer).

D. Feature Engineering

Our approach inherits the limitations of IRL approaches that require specifying reward features by hand. Both features and costs depend on several parameters which require tuning. The problem becomes especially difficult as our features simultaneously govern the behaviour of the reward learning as well as the trajectory optimizer. For instance, high gradients in the feature function lead to erratic behaviour of the optimizer leading to poor solutions and convergence to local minima. 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 (3).

A crucial aspect of feature engineering is defining the context as it determines how expressive the features are. We have considered a limited set of vectors as context in this work (i.e. obstacle position, start and goal positions). However, it is possible to include additional information in the context such as object properties (e.g. sharp, fragile or liquid) [1], human position [34], and 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 can lead to increased training time. An evaluation of the trade-off between improved generalisation and higher training time would be interesting, but is out of the scope of this study and we leave it for future work.

VI. Conclusion and Future Work

We presented a new approach for learning and executing human preferences in manipulation tasks. Our simulation studies showed the fast convergence of the algorithm, as well as the proof-of-concept for generalising path and velocity preferences. We further validated the efficiency and accuracy of our approach in a user study in a real world scenario. Our results suggest that user defined models can contribute to planning of manipulation tasks through reduction of interaction effort and achieving personalized trajectories.

Using a trajectory model allowed us to make the problem tractable, however this model is quite simplistic and does not describe human motion behaviour very well. Future research could consider replacing 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 to contact with each other through a shared object (physical human-robot collaboration). We believe the framework is specially effective in collaborative settings where knowledge of preferences of a partner are essential to execution of the task. Finally, it is of interest to study whether more complex non-linear formulations of the reward function using Gaussian Process [35] or Neural Networks [16] can effectively capture such preferences without the need for rigorous feature engineering.

References

[1] A. Jain, S. Sharma, T. Joachims, and A. Saxena, "Learning preferences for manipulation tasks from online coactive feedback," *The Int. J. of Robotics Research*, vol. 34, no. 10, pp. 1296–1313, 2015.

[2] L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez, and C. Torras, "Learning physical collaborative robot behaviors from human demonstrations," *IEEE Trans. on Robotics*, vol. 32, no. 3, pp. 513–527, 2016.

[3] R. Ikeura and H. Inooka, "Variable impedance control of a robot for cooperation with a human," in *Proceedings of 1995 IEEE Int. Conf. on Robotics and Automation*, IEEE, vol. 3, 1995, pp. 3097–3102.

[4] T. Tsumugiwa, R. Yokogawa, and K. Hara, "Variable impedance control based on estimation of human arm stiffness for human-robot cooperative calligraphic task," in *Proceedings 2002 IEEE Int. Conf. on Robotics and Automation (Cat. No. 02CH37292)*, IEEE, vol. 1, 2002, pp. 644–650.

[5] V. Duchaine and C. M. Gosselin, "General model of human-robot cooperation using a novel velocity based variable impedance control," in *Second Joint EuroHaptics Conf. and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'07)*, IEEE, 2007, pp. 446–451.

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

[7] L. Peternel, N. Tsagarakis, and A. Ajoudani, "A human–robot co-manipulation approach based on human sensorimotor information," *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 7, pp. 811–822, 2017.

[8] T. Osa, A. M. G. Esfahani, R. Stolkin, R. Lioutikov, J. Peters, and G. Neumann, "Guiding trajectory optimization by demonstrated distributions," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 819–826, 2017.

[9] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings 2002 IEEE Int. Conf. on Robotics and Automation (Cat. No. 02CH37292)*, IEEE, vol. 2, 2002, pp. 1398–1403.

[10] T. Kulvicius, M. Biehl, M. J. Aein, M. Tamosiunaite, and F. Wörgötter, "Interaction learning for dynamic movement primitives used in cooperative robotic tasks," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1450–1459, 2013.

[11] A. Gams, B. Nemec, A. J. Ijspeert, and A. Ude, "Coupling movement primitives: Interaction with the environment and bimanual tasks," *IEEE Trans. on Robotics*, vol. 30, no. 4, pp. 816–830, 2014.

[12] L. Peternel, T. Petrič, E. Oztop, and J. Babič, "Teaching robots to cooperate with humans in dynamic manipulation tasks based on multi-modal human-in-the-loop approach," *Autonomous robots*, vol. 36, no. 1, pp. 123–136, 2014.

[13] B. Nemec, N. Likar, A. Gams, and A. Ude, "Human robot cooperation with compliance adaptation along the motion trajectory," *Autonomous robots*, vol. 42, no. 5, pp. 1023–1035, 2018.

[14] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning.," in *Icml*, vol. 1, 2000, p. 2.

[15] C. Wirth, R. Akrour, G. Neumann, J. Fürnkranz, *et al.*, "A survey of preference-based reinforcement learning methods," *J. of Machine Learning Research*, vol. 18, no. 136, pp. 1–46, 2017.

[16] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei, "Reward learning from human preferences and demonstrations in atari," *arXiv preprint arXiv:1811.06521*, 2018.

[17] P. Shivaswamy and T. Joachims, "Coactive learning," *J. of Artificial Intelligence Research*, vol. 53, pp. 1–40, 2015.

[18] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proceedings of the 23rd Int. Conf. on Machine learning*, 2006, pp. 729–736.

[19] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, *et al.*, "Maximum entropy inverse reinforcement learning.," in *Aaai*, Chicago, IL, USA, vol. 8, 2008, pp. 1433–1438.

[20] M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh, "Learning reward functions by integrating human demonstrations and preferences," *arXiv preprint arXiv:1906.08928*, 2019.

[21] M. Fahad, Z. Chen, and Y. Guo, "Learning how pedestrians navigate: A deep inverse reinforcement learning approach," in *2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 819–826.

[22] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The Int. J. of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[23] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: An experimental comparison," in *2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, IEEE, 2014, pp. 1341–1346.

[24] MathWorks. (2018). "Waypoint trajectory generator," [Online]. Available: https://nl.mathworks.com/help/nav/ref/waypointtrajectory-system-object.html (visited on 08/10/2021).

[25] A. Albu-Schaffer, C. Ott, U. Frese, and G. Hirzinger, "Cartesian impedance control of redundant robots: Recent results with the dlr-light-weight-arms," in *2003 IEEE Int. Conf. on robotics and automation*, IEEE, vol. 3, 2003, pp. 3704–3709.

[26] D. Sadigh, A. D. Dragan, S. Sastry, and S. A. Seshia, "Active preference-based learning of reward functions," 2017.

[27] S. Calinon and D. Lee, "Learning control," in *Humanoid Robotics: a Reference*, P. Vadakkepat and A. Goswami, Eds., Springer, 2019, pp. 1–52. DOI: 10.1007/978-94-007-7194-9_68-1.

[28] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*, Springer, 1986, pp. 396–404.

[29] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Advances in psychology*, vol. 52, Elsevier, 1988, pp. 139–183.

[30] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, "Keyframe-based learning from demonstration," *Int. J. of Social Robotics*, vol. 4, no. 4, pp. 343–355, 2012.

[31] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The Int. J. of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[32] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in *2010 IEEE/RSJ Int. Conf. on intelligent robots and systems*, IEEE, 2010, pp. 3232–3237.

[33] F. Flemisch, D. A. Abbink, M. Itoh, M.-P. Pacaux-Lemoine, and G. Weßel, "Joining the blunt and the pointy end of the spear: Towards a common framework of joint action, human–machine cooperation, cooperative guidance and control, shared, traded and supervisory control," *Cognition, Technology & Work*, vol. 21, no. 4, pp. 555–568, 2019.

[34] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan, "Learning from physical human corrections, one feature at a time," in *Proceedings of the 2018 ACM/IEEE Int. Conf. on Human-Robot Interaction*, 2018, pp. 141–149.

[35] E. Bıyık, N. Huynh, M. J. Kochenderfer, and D. Sadigh, "Active preference-based gaussian process regression for reward learning," *arXiv preprint arXiv:2005.02575*, 2020.

A

Appendix - Feature Design and Cost Formulation

Features are computed from robot's state and context and are used to describe human preferences. They simultaneously govern the behaviour of the learning algorithm, as well as the trajectory optimization. Cost functions are defined in the same manner, but they only influence the behaviour of the optimization. The weights associated with the features are learned from demonstration via the IRL algorithm discussed in Chapter 1, but the weights associated with the costs are hand-tuned.

A.1. Human Features

We employed 3 path features and 18 velocity features in our framework, all which had weights assigned to based on human demonstrations. The 3 path features were unique and of different nature, while the 18 velocity features were all the RBFs with different centers. As these features are employed in an optimization algorithm, we payed attention to defining them as smooth as possible to avoid undesired behaviours such getting stuck in local minima.

A.1.1. Distance to Obstacle

For this feature we use an exponential function with a negative exponent as shown in A.1). We use a simple radial definition for the distance to the obstacle, measured from the center of the obstacle. This assumes that the shape of the obstacle is spherical. Although this is usually not the case, we can approximate the obstacle by the smallest sphere that it can fit within. The feature ϕ_d can then be defined as

$$\phi_d = e^{-\beta d^2}. \tag{A.1}$$

where d is the euclidean distance to the center of the obstacle, and β is a parameter defining the shape of the function.

In our experiments, we considered obstacles of about 12 cm wide (radius of 6 cm). If the size of the obstacle is increased beyond this value, re-tuning of features is required (i.e. we do not generalize to larger obstacle sizes).

As seen in Fig. A.1, the feature value drops to 0 at 30 cm away from the obstacle. This is a threshold outside of which the behaviour of the trajectory is no longer affected by the distance to the obstacle. Note that if a negative weight is learned associated with this feature, the trajectory is still attracted towards the obstacle even if it initially lies outside of the 30 cm threshold. This is because our optimization strategy 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.

A.1.2. Obstacle Side

For this feature we initially have to find out which side of the obstacle a trajectory sample lies on. To this extent, we first define the plane that lies in the middle of the obstacle, is parallel to the axis that

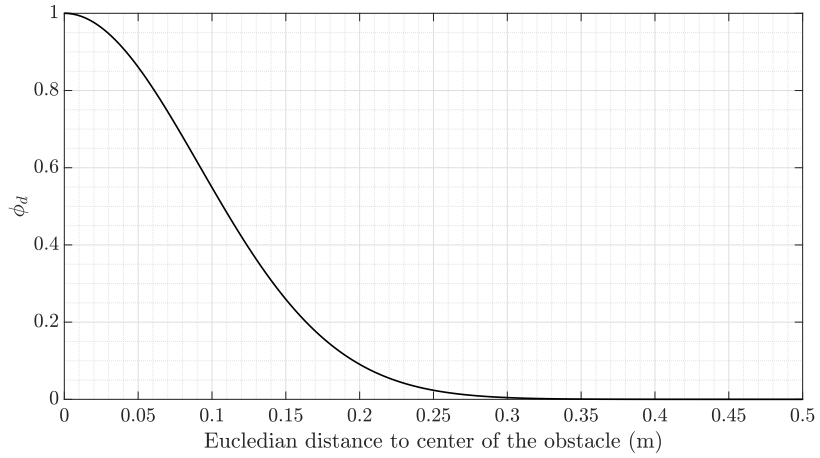


Figure A.1. Exponential feature for learning the preferred distance to the obstacle. Based on the sign of the weight, we learn to either attract the trajectory towards the obstacle or repel from it.

defines the direction of the motion, and is perpendicular to the horizontal plane (see Fig. A.2). Hence, we find three points that lie on this plane based on the position of the obstacle and the direction of the motion (This is defined as the line passing through the center of the obstacle and parallel to the longest axis of the trajectory). Lets denote these points as $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ (order is not important). We can then use the methodology in [1] to determine which side of the plane a trajectory sample lies on. Accordingly, we define the following differences: $\mathbf{B}' = \mathbf{B} - \mathbf{A}$, $\mathbf{C}' = \mathbf{C} - \mathbf{A}$, and $\mathbf{X} = \mathbf{x} - \mathbf{A}$, where $\mathbf{x}$ is the trajectory sample position. Finally, the sign of $S = \mathbf{det}([\mathbf{B}'; \mathbf{C}'; \mathbf{X}'])$ determines which side of the plane the trajectory sample lies on. Importantly, the magnitude of S also determines how far to each side of the plane a sample point lies (linear relation).

While it is possible to directly use the lateral distance of each point to the obstacle's center, this method allows for generalisation to scenarios where the plane is not parallel to the reference axis. For instance, the plane can be defined parallel to the line connecting the start and goal position. Therefore, we recommend the above formulation.

We then define the feature with a tangent hyperbolic function given by

$$\phi_s = \frac{2}{1 + e^{\gamma S}} - 1, \tag{A.2}$$

where γ defines the shape of the function. It should be noted that the sign of γ depends on the direction of trajectory (i.e. we flip the function in case of planning in the opposite direction to stay consistent with the definition of 'close'/'far' side of the obstacle).

The resulting function can be seen in Fig. A.3. This feature spans over the entire workspace as we want it to capture exactly how far to each side of the obstacle the user prefers the trajectory to be. While the trajectory can never reach as far as 1 meter to one side of the obstacle due to the workspace restrictions, we found the function with this shape to perform better compared to functions with higher gradients that fit within a smaller distance range.

A.1.3. Height from the Table

To encode this preference, we designed a sigmoid function with the vertical distance of a trajectory sample from the table h as the input:

$$\phi_h = \frac{1}{1 + e^{-\lambda(h+p)}}. \tag{A.3}$$

Here, λ defines the shape of the function and p determines the center of the function. The resulting function is shown in Fig. A.4, and we can see that the slope gradually decreases at heights close to the upper and lower height boundaries of the workspace which are respectively at 5 cm and 80 cm. This is done to hinder the effect of a change in height on both the weight update and the optimization close to these boundaries (i.e. A demonstration at 75cm above the table should not impact the weight

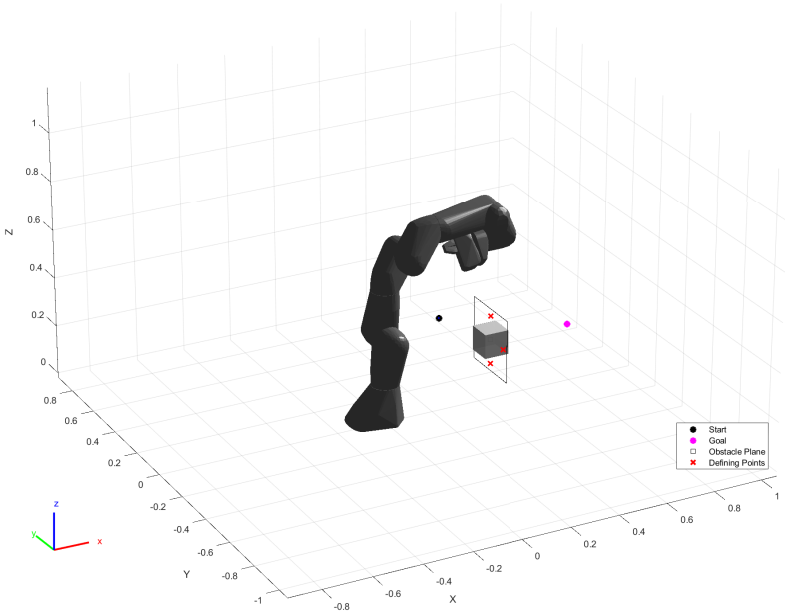


Figure A.2. The definition of the plane of the obstacle parallel to the longest axis of the trajectory. We define and use the three red points on this plane to find out if trajectory samples lie on the close or far side of the obstacle.

update very differently to a demonstration at a height of 70cm). This gives the freedom to the other objectives that are active in such areas (e.g. side feature) to have a higher impact on the optimization. The algorithm learns either a positive or a negative weight related to this feature which can respectively pull the trajectory to the table or push it away.

A.1.4. Velocity Features

The velocity features differ from the rest of the features in that they are, naturally, defined in the velocity space. These features share the same structure based on RBFs, and are evenly distributed over a range of velocities as shown in Fig. A.5. Each feature can be defined by the following expression:

$$\psi_i(\dot{x}) = e^{-(\varepsilon\dot{x}-c_i)^2}, \tag{A.4}$$

where $\dot{x}$ is the average velocity for a given trajectory segment, c_i is the velocity the i^{th} function is centered at, and ε is the variable defining the shape of the functions. Hence, the velocity feature vector can be defined as $\boldsymbol{\phi}_v = [\psi_1, \psi_2, \cdots, \psi_n]$

We discretize these features in to two regions of the workspace: The region considered to be 'close' to the obstacle, and the region considered to be 'far' from the obstacle. The definition for 'close' and 'far' is based on a distance threshold δ. If the center of a segment lies within a distance δ from the center of the obstacle, we consider it to be 'close' and featurize its velocity in the vector $\boldsymbol{\phi}_{v1}$, and otherwise in $\boldsymbol{\phi}_{v2}$. Therefore, we have in total $2i$ velocity features. As a result, in the optimization, the optimal velocity is computed based on the distance to the obstacle and the corresponding feature vector.

A.2. Robot Objectives

We employ 2 costs in the path optimization and 1 reward in the velocity optimization which define the behaviour of the robot in absence of any human learned weights. These objectives are essential in counter-balancing the behaviours that emerge from the human features in A.1.

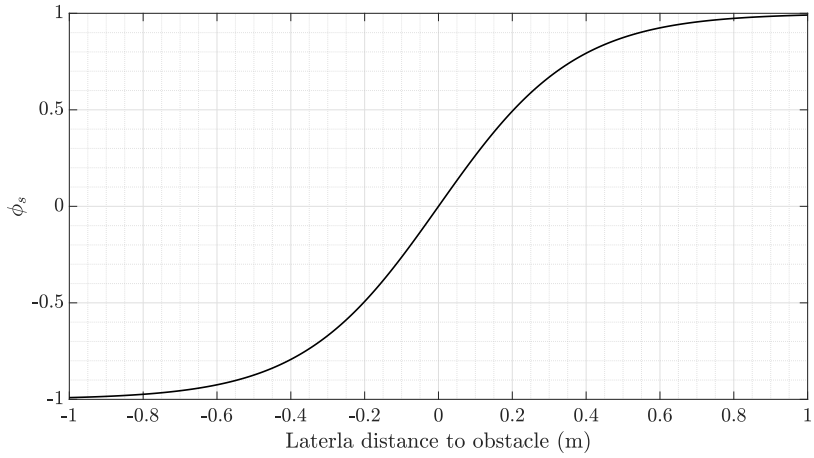


Figure A.3. Tangent hyperbolic function designed for the side feature. The large span of the function means that the function is active in all regions of the workspace. However, as the gradient of this function decreases at larger lateral distances, so does the influence of this function in the trajectory optimization.

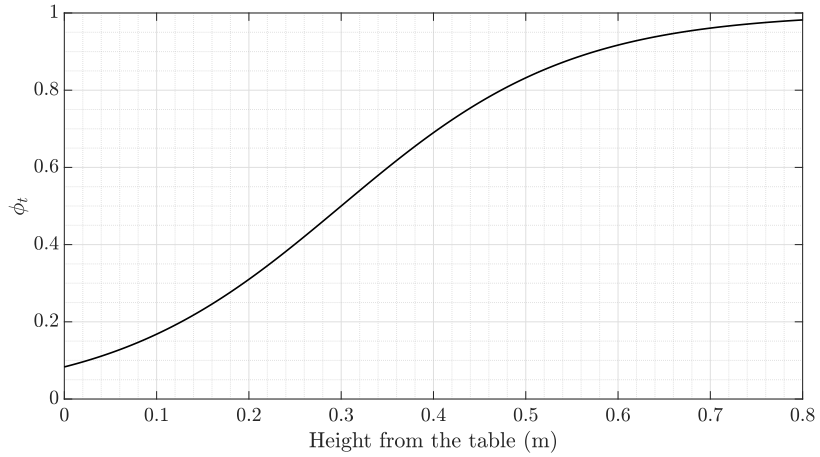


Figure A.4. Sigmoid function defining the behaviour of the height from the table preference. Ideally the function should saturate close to 0 cm from the table, but this would mean a higher gradient at heights in the middle range. We found a lower gradient to improve the optimization behaviour with respect to counter balancing the other features.

A.2.1. Path Efficiency Cost

This cost is simply computed as the sum of Eucledian distances between every trajectory sample, which add up to the total length of a trajectory

$$J_E = \sum_{k=1}^{N-1} \left\| x(k+1) - x(k) \right\|_2 . \tag{A.5}$$

The trajectory with the lowest J_E is thus the straight line between the start and the goal positions. While simple, this cost is critical as it moderates the behaviour of all human path features. In the absence of this cost, the optimized waypoint p_m^* that defines the path would always converge to the boundaries defined in (6). For instance, a negative weight for the height feature would always push the trajectory to the highest point, no matter the magnitude. As a result, the weight associated with this cost has to be finely tuned.

A.2.2. Collision Avoidance Cost

As mentioned in A.1.2, the obstacle side feature can result in the trajectory being pulled towards the obstacle. Furthermore, the path efficiency cost could also attract the trajectory towards an obstacle

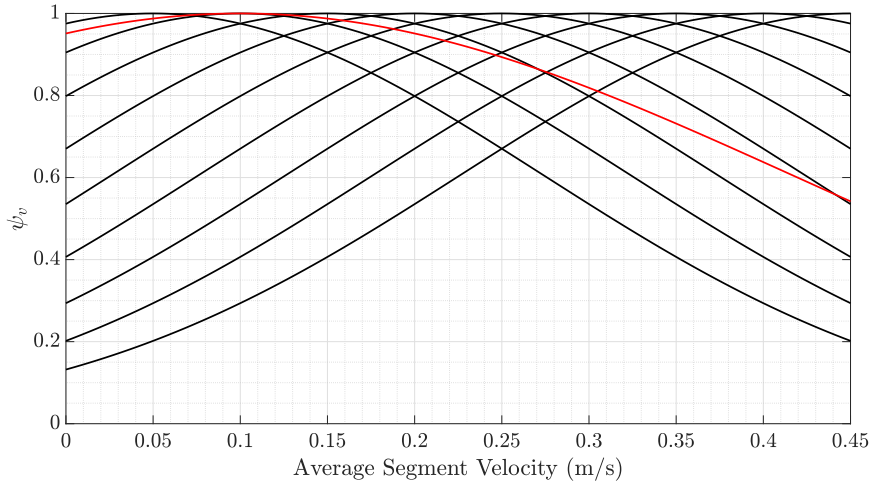


Figure A.5. Velocity features (black: human, red: robot) defined over a range of velocities. Depending on the weight learned by the algorithm, the features in black are stretched vertically to represent the corresponding reward/cost associated with each velocity. The feature in red however is always multiplied by the same **negative** weight to encourage the convergence of the optimization towards this velocity.

if the obstacle is placed close to the shortest path. As a result, we need a high cost associated with trajectory samples that lie close enough to the obstacle to cause a collision. For this we use the following collision cost formulation from [2]:

$$J_c(\mathbf{x}) = \begin{cases} -D(\mathbf{x}) + \frac{1}{2}\epsilon & \text{if } D(\mathbf{x}) < 0 \\ \frac{1}{2\epsilon}(D(\mathbf{x}) - \epsilon)^2 & \text{if } 0 \leq D(\mathbf{x}) \leq \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{A.6}$$

where $D(\mathbf{x})$ is the distance of a sample from the **surface** of the obstacle, ϵ is the distance threshold from which the cost activates (i.e. has a non-zero value). Fig. A.6 illustrates J_c, which has a high gradient once the sample distance to the obstacle drops below the threshold.

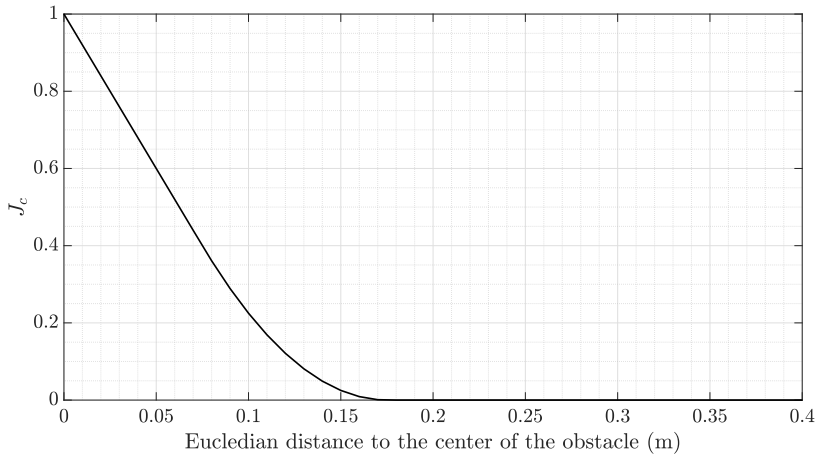


Figure A.6. Collision cost as a function of distance to the center of the obstacle. Note that the function is smooth, and has no impact on the trajectory outside of the threshold ϵ (here $\epsilon = 0.15$).

A.2.3. Slow Velocity Reward

This reward is designed in to encourage manipulations at lower velocities. In absence of any human demonstrations ($\boldsymbol{\theta}_{HV} = 0$), the robot should plan the task in a safe and slow manner. However, humans

can override this safety measure by providing demonstrations at higher velocities. The function employs the same RBF structure as in (A.4), but with a different shape factor ε to increase its span over the velocity space (see Fig. A.5).

B

Appendix - Simulation and User Study Details

In this chapter we provide practical details and additional results obtained from the two studies discussed in the paper. Furthermore, we cover additional theory for the methods used in these studies.

B.1. Contexts and Trajectories of the Convergence Test

In Sec. III-A, we discussed the convergence of path preference weights θ_{HP}. However, these weights alone do not define the resulting trajectory from the framework, as the robot objectives and the trajectory space used in the optimization also influence the output. This gives rise to situations where there is a discrepancy between the convergence of the weights and convergence of the trajectory. As a result, it is worth evaluating the resulting trajectory from the 10 tasks studied in Sec. III-A.

Fig. B.1a. illustrates the resulting trajectories after 5 iterations of updating the weights from providing the same demonstration (red trajectory). We tested the convergence in 5 different contexts (i.e. start, goal, and obstacle positions), each with two different set of path preferences. Comparing the results with the weight convergence curves in Fig. 4, we can see that failure to fully converge in weights does not necessary mean failure in converging to the demonstrated trajectory. An example of this is scenarios c. Conversely, near-full convergence of weights also does not always result in accurate reproduction of the trajectory provided as feedback. This can be seen in scenarios b and d. The behaviour of the optimizer indicates that it is more sensitive to small differences in weight distribution in certain regions of the space than others. Even though in both scenarios b and d the alignment factor m reaches a value close to 1, the weights are still slightly different than those in θ_{HP}.

The results shown in Sec. IV showed that such small errors in the reproduced trajectory do not significantly affect the subjective satisfaction of users with the results. In general, our framework is designed to produce trajectories that come close to those demonstrated by the user, and more importantly adhere to the preferences. However, combination of the framework with other imitation learning algorithms that achieve high accuracy of motion on lower levels could be a promising direction of research in the future to improve the behaviour.

B.2. Proof of Concept Study and Dynamic Movement Primitives

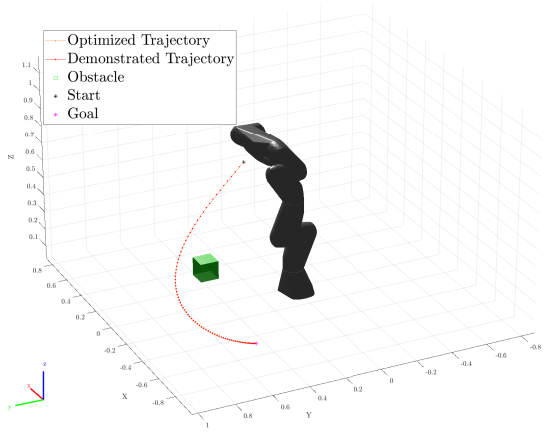
To compliment the discussion and results in Sec. III-B, we begin by providing a more detailed description of DMPs in this section. We use the DMPs formulation from [3], where for a discrete movement (point-to-point), acceleration $\ddot{\boldsymbol{x}}$ is modulated by equations

$$\ddot{\boldsymbol{x}} = k_p(\boldsymbol{g} - \boldsymbol{x}) + k_v\dot{\boldsymbol{x}} + \mathbf{f}(s), \tag{B.1a}$$

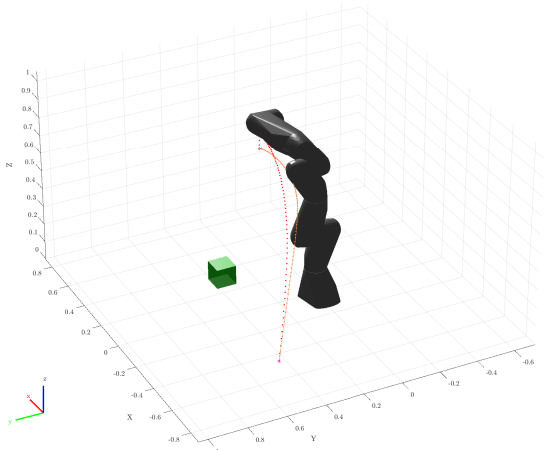
$$\dot{s} = \alpha_s s, \tag{B.1b}$$

where $\boldsymbol{g}$ is the attractor point (goal position in our case), k_p and k_v are the stiffness and damping factors (we use ideal underdamped behaviour), s is a sigmoidal decaying phase variable (starts from 1 and

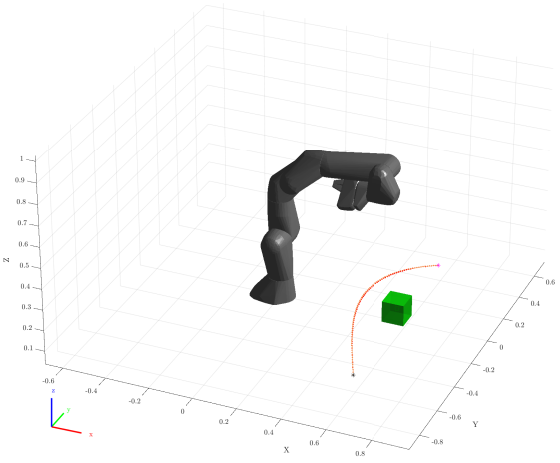
converges to 0 by the end of the motion) with α_s as the decaying factor, and $\mathbf{f}(s)$ is the non-linear



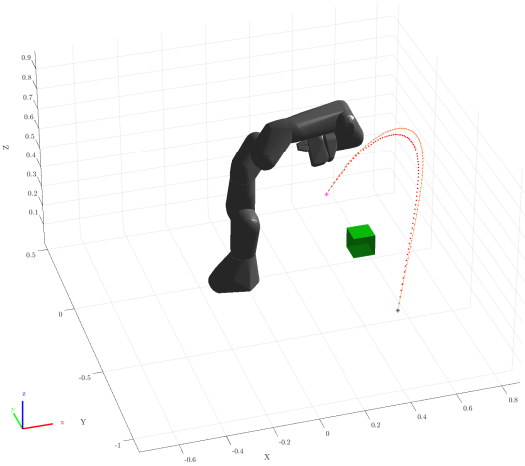
(a)



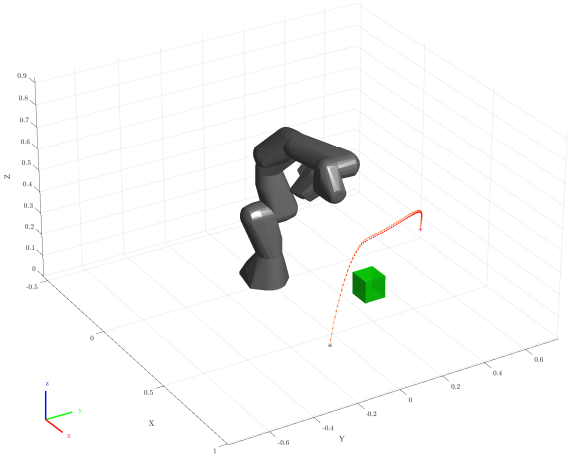
(b)



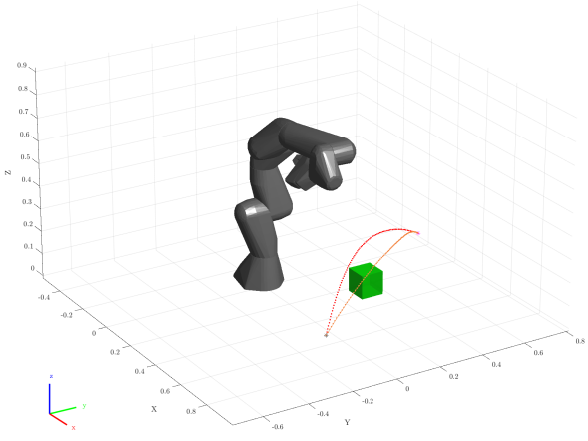
(c)



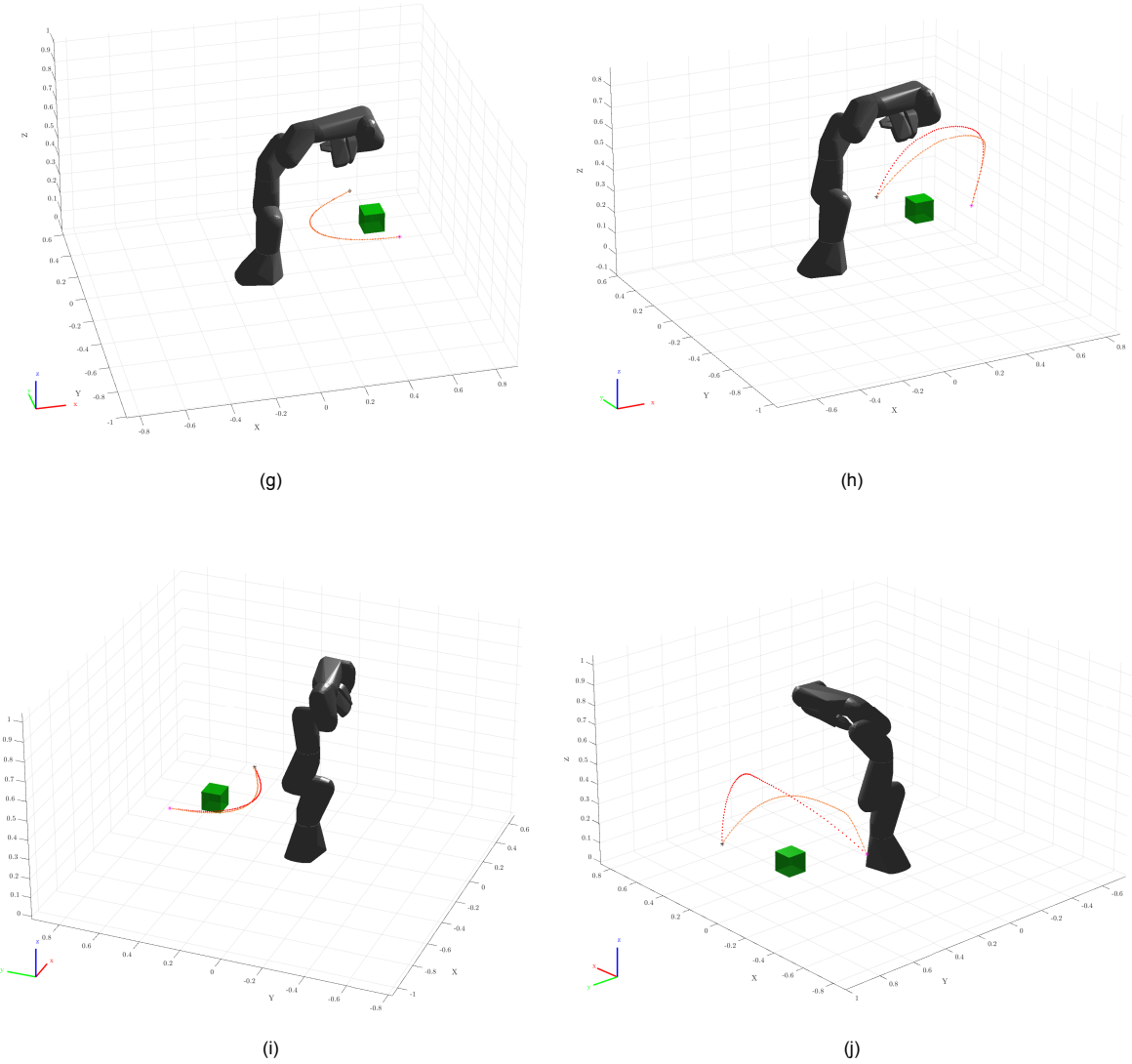
(d)



(e)



(f)



(g) (h)

(i) (j)

Figure B.1. The context and trajectories from a-j respectively correspond to the curves in Fig. 4. Each row of figures shows the same context, but for two different path preferences. The preference optimized results (orange) are obtained after 5 iterations of feedback.

forcing term that modulates the trajectory shape. The forcing term progressively disappears and lets the spring-damper system drive the behaviour to convergence to $\boldsymbol{g}$ [4]. In our case, the forcing term $\mathbf{f}(s)$ is given by a linear combination of RBFs

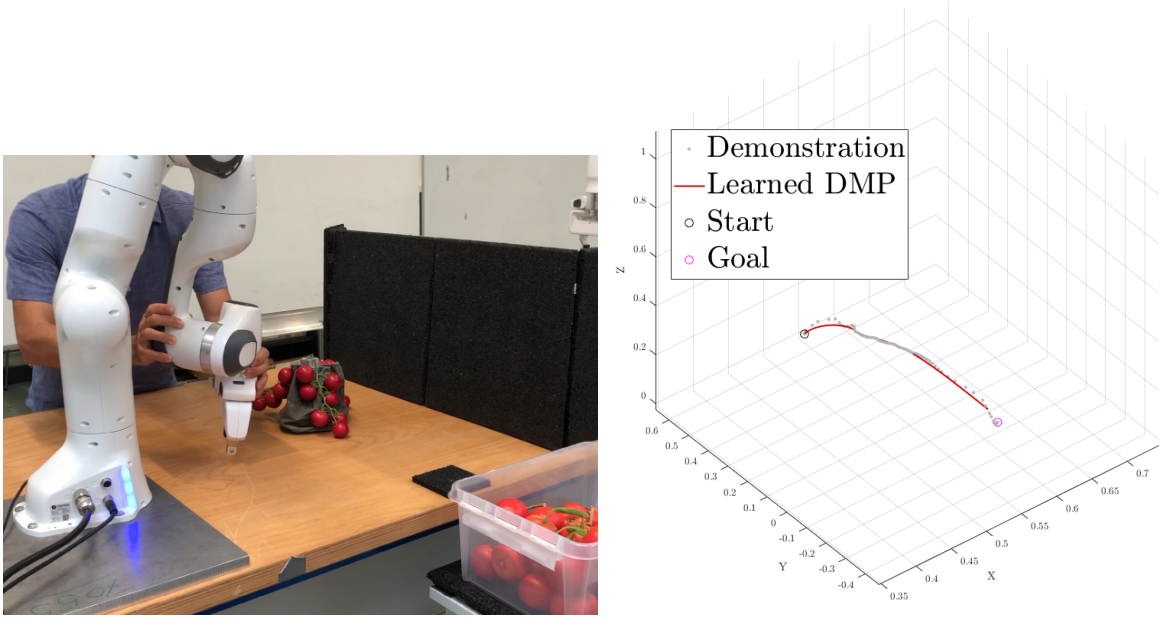
$$\mathbf{f}(s) = \frac{\sum_{i=1}^{N} \Psi_i(s) w_i}{\sum_{i=1}^{N} \Psi_i(s)} s, \tag{B.2}$$

where $\Psi_i(s)$ are fixed basis functions and w_i are adjustable weights. We use locally weighted regression for training the weight parameters.

In order to equip the DMPs with obstacle avoidance capability during the reproduction phase, we use the repulsive potential gradient term from [5] in the $x-y$ plane

$$\nabla \boldsymbol{U}_{\text{rep}}(\boldsymbol{x}) = \begin{cases} k_o \left(\frac{1}{\tau} - \frac{1}{D(\boldsymbol{x})} \right) \frac{1}{D^2(\boldsymbol{x})} \nabla D(\boldsymbol{x}), & D(\boldsymbol{x}) \leq \tau \\ 0, & D(\boldsymbol{x}) > \tau, \end{cases} \tag{B.3}$$

where τ is a distance threshold outside of which the field had no influence, k_o is the stiffness term that



(a) (b)

Figure B.2. (a) The setup for the proof of concept experiment. The human prefers the robot to perform manipulations on the close side of the obstacle to avoid risk of collision with the black wall on the other side. Furthermore, in the position shown, the user demonstrates a slow velocity, informing the robot of the preference of slowing down when close to the obstacle. (b) The demonstration provided to the DMPs and the resulting encoded trajectory. This demonstration was provided in the same setting as (a), with the exception that the obstacle was removed. Note the higher density of gray points at the center shows that the user slowed down in the middle of the trajectory.

defines the strength of the repulsive field, and $D(\boldsymbol{x})$ indicates the Euclidean distance to the center of the obstacle in the $x-y$ plane. Note that we set the third component of $\nabla \boldsymbol{U}_{\text{rep}}$ to 0. We add this term to (B.1a) as an additional forcing term that pushes the trajectory away from the obstacle

$$\ddot{\boldsymbol{x}} = k_p(\boldsymbol{g} - \boldsymbol{x}) + k_v\dot{\boldsymbol{x}} + \mathbf{f}(s) + \nabla \boldsymbol{U}_{\text{rep}}(\boldsymbol{x}). \tag{B.4}$$

Fig. B.2 illustrates the demonstrations provided to our framework and the DMPs. DMPs do not require previous knowledge of the position of the obstacle in order to later produce collison-free trajectories. Nevertheless, this lack of knowledge is the underlying cause of the inability of this framework in generalising the preference of staying on the closer side to different start, goal and obstacle positions. Note that although we provided a demonstration to the DMPs where we slow down in the middle of the trajectory, this is an unfair advantage because the model normally does not have the information that the obstacle is located around the middle are of the trajectory.

B.3. User Study Details

B.3.1. Modifications and Self-collision Avoidance

In this section, we first address the modification made to our methodology for learning and optimizing of velocity preferences in the first user study experiment. We simplify the learning process to obtaining the cumulative feature vectors Φ_{V2} and Φ_{V2} from the demonstration via (2b) (i.e. no weights or reward functions calculated). The approximation of the preferred speed of motion within each distance bin then simply amounts to sorting the vectors by the highest cumulative feature count:

$$\dot{x}^*(r,d) = \arg\max_{c_j} \Phi_V. \tag{B.5}$$

Here c_j refers to the center of the RBFs described in (4). Note that $\Phi_V = \{\Phi_{V2}, \Phi_{V2}\}$ and covers both the “close” and “far” distance regions. Using this method we essentially discretise the desired velocity to the centers of the RBFs shown in Fig. A.5 based on the demonstrations. The absence of weights makes this a one-shot learning algorithm, as there is no mechanism to update the estimate. Compared

to (7), (B.5) can be solved much quicker, which was needed for making the user study viable. However, with this method robot objectives or additional velocity preferences can not be addressed, and velocity values between the centers of the RBFs c_j can not be achieved. Hence, it is beneficial in future studies to investigate how the method using optimization can be performed faster.

An additional scheme that was applied during the experiments was a feasibility function that worked as a post-filtering step to assure trajectories were not falling into self-collision or joint limits. This was especially helpful in a case where a user provided a demonstration on the side of the obstacle closer to the robot, which when generalised to another task was resulting in a self-collision warning. In this function we first converted the robot's trajectory ξ_R into a sequence of robot configurations $\mathbf{q}_R$ using the generalized inverse kinematics tool from [6]. If the tool is given a few centimeters of tolerance on the end-effector position for solving the inverse kinematics, it automatically generates configurations that are safe.

Having found the safe sequence of robot configurations $\mathbf{q}_R$, we still need to find out the new safe trajectory in the task space, since our impedance controller is defined within this space. Therefore, we perform a forward kinematics sequence back to the Cartesian coordinates by first computing the 4×4 homogeneous transformation matrix $\mathbf{T}$ from the end-effector frame to the robot's base frame for every configuration in $\mathbf{q}_R$. Taking the base frame as the origin of the task space, we extract the translation vector from every $\mathbf{T}$ to obtain the Cartesian coordinates of our safe trajectory.

B.3.2. Additional Results from Experiment 2

In this section we present further results from the experiment described in Sec. IV-B. The black circle in the plot represents the obstacle. These results compare the resulting trajectories from our framework with the ground truth demonstrations in Task 3, and are shown in Fig. B.3. Even though we instructed the users to demonstrate the same set of preferences, we notice that different users exhibit different behaviours in terms of "'distance to the obstacle" and "obstacle side" preference. For instance, user 4 stays significantly closer to the obstacle and more towards the middle compared to user 6. It is evident that our framework can capture such differences in amplitudes of the same preference, as the resulting trajectories (shown in orange) remain close to the demonstrated trajectories (shown in blue).

We can see that the result is not always accurate in terms of the shape of trajectory, nor do the results always lie within the uncertainty region demonstrated by the users (yellow area). As discussed in Sec. V-C, this is due to the fact that we are parametrizing the entire task space using only a few parameters. Hence, our method does not achieve fully accurate imitation of movements. Nevertheless, this is not the main objective of the method, and we observe that in terms of the adhering to the preferences demonstrated, the robot performs well.

We notice that there is not a significant correlation between the accuracy of the trajectory and whether it was the result of training on the same task or generalization from training on another task instance. This indicates that our method is robust to learning in different context. Another point to note is that while we resample the trajectories to extract an evenly distributed path for the learning stage, there remain artifacts in the blue trajectory with density of samples higher in certain regions. This due to the fact that some users perform demonstrations with inconsistent velocities which creates practical issues for resampling of the trajectory. Such inconsistencies can affect the performance of the learning algorithm as they affect the feature values used for the weight update. In future work better resampling strategies might improve the performance of the framework.

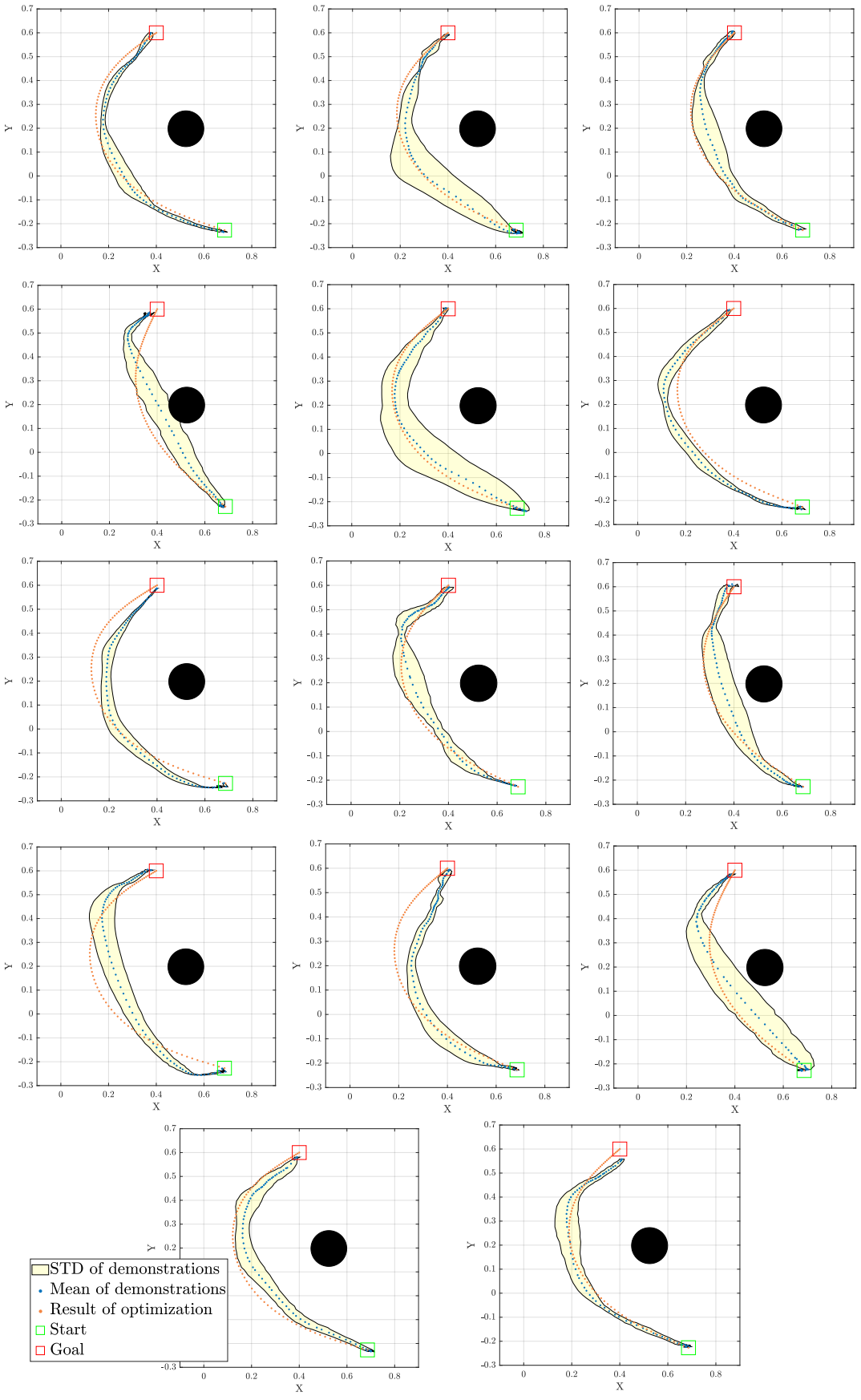


Figure B.3. Demonstrated and optimized trajectories seen from above ($x-y$ plane) in Task 3. Starting from the top left plot we label the plots from 1-14 going through the rows. Plots with an odd number are result of generalization (model trained on Task 1), and plots with even number show results from learning preferences in the same task.

Bibliography

[1] Harald Hanche-Olsen (https://math.stackexchange.com/users/23290/harald-hanche-olsen). *Side of a plane in 3D space*. Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/214194 (version: 2012-10-15). eprint: https://math.stackexchange.com/q/214194. URL: https://math.stackexchange.com/q/214194.

[2] Matt Zucker et al. “Chomp: Covariant hamiltonian optimization for motion planning”. In: *The International Journal of Robotics Research* 32.9-10 (2013), pp. 1164–1193.

[3] S. Calinon and D. Lee. “Learning Control”. In: *Humanoid Robotics: a Reference*. Ed. by P. Vadakkepat and A. Goswami. Springer, 2019, pp. 1–52. DOI: 10.1007/978-94-007-7194-9_68-1.

[4] Matteo Saveriano et al. “Dynamic Movement Primitives in Robotics: A Tutorial Survey”. In: *arXiv preprint arXiv:2102.03861* (2021).

[5] Ji Lee, G.D. Hager, and Z. Dodds. *Robotic Motion Planning: Potential Functions*. URL: https://www.cs.cmu.edu/~motionplanning/lecture/Chap4-Potential-Field_howie.pdf (visited on 08/29/2021).

[6] MathWorks. *generalized Inverse Kinematics*. 2017. URL: https://nl.mathworks.com/help/robotics/ref/generalizedinversekinematics-system-object.html (visited on 09/09/2021).